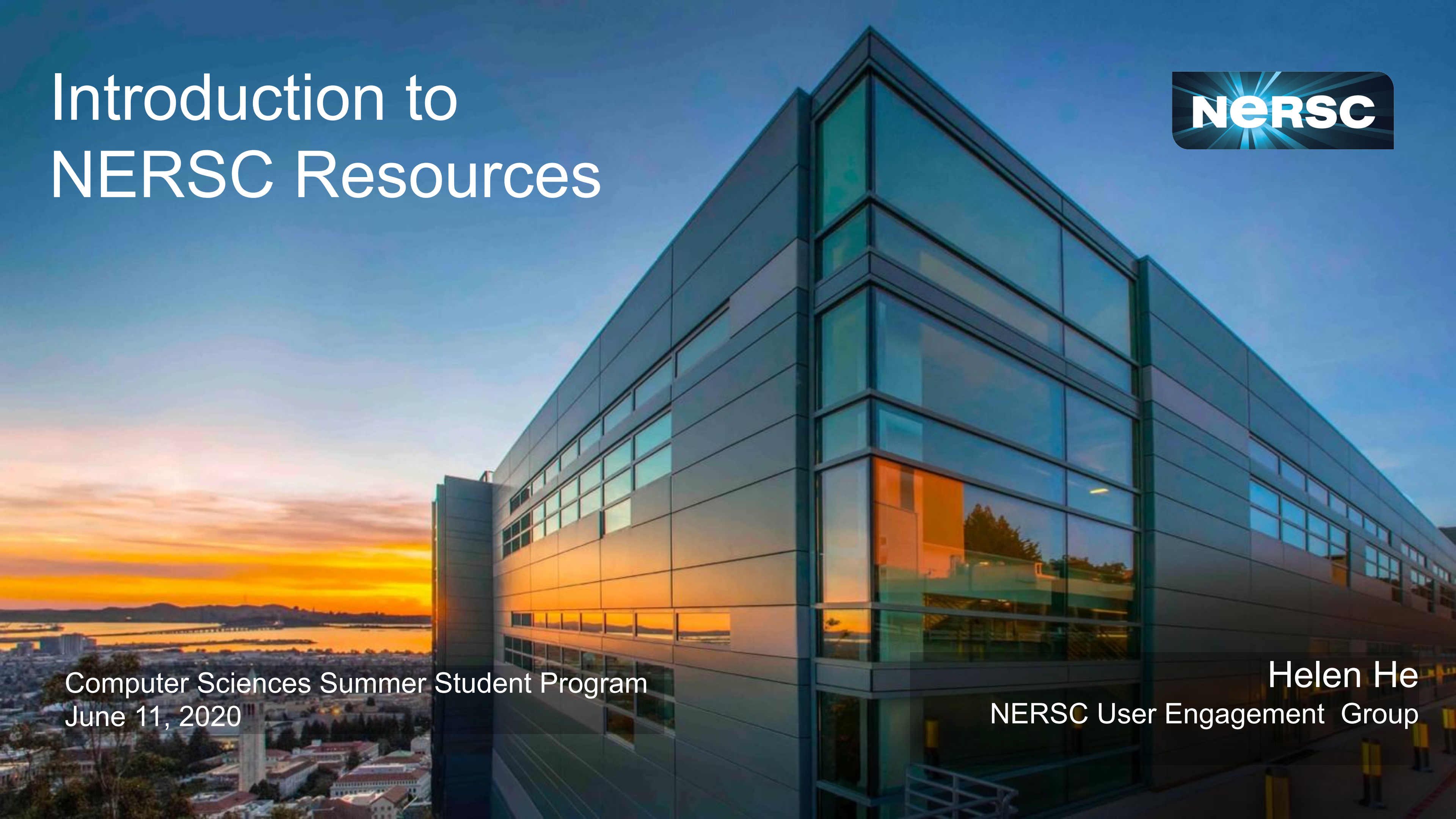


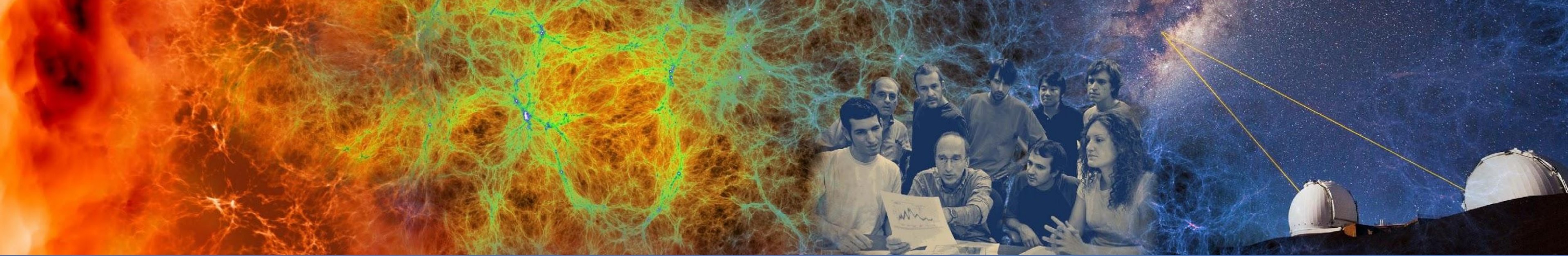
Introduction to NERSC Resources

Computer Sciences Summer Student Program
June 11, 2020

Helen He
NERSC User Engagement Group

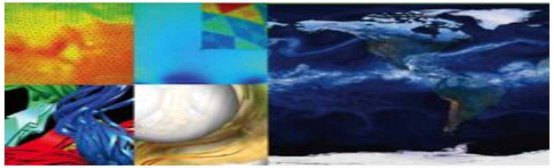
Outline

- NERSC and Systems Overview
- Connecting to NERSC
- File Systems and Data Management/Transfer
- Software Environment / Building Applications
- Running Jobs
- Data Analytics Software and Services
- NERSC Online Resources
- Hands-on: Compiling and Running Jobs

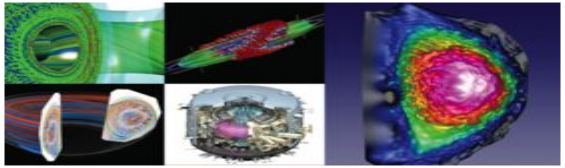


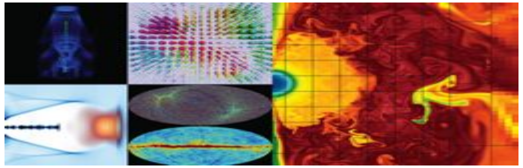
NERSC and Systems Overview

NERSC is the Mission HPC Computing Center for the DOE Office of Science



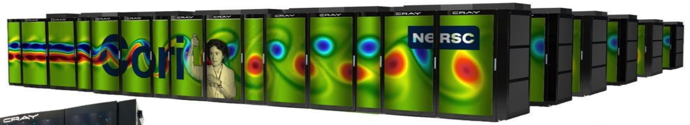
- NERSC deploys advanced HPC and data systems for the broad Office of Science community
- NERSC staff provide advanced application and system performance expertise to users
- Approximately 7,000 users and 800 projects
- Over 2,000 publications cite using NERSC resources per year
- Founded in 1974, focused on open science
- Division of Lawrence Berkeley National Laboratory

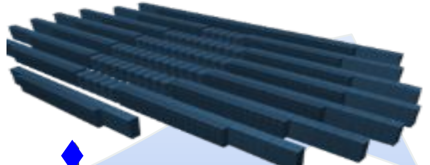




ASCR	Advanced Scientific Computing Research
BER	Biological & Environmental Research
BES	Basic Energy Sciences
FES	Fusion Energy Sciences
HEP	High Energy Physics
NP	Nuclear Physics
SBIR	Small Business Innovation Research

NERSC Systems Roadmap





2013

NERSC-7: Edison
Multi-core CPU

2016

NERSC-8: Cori
Manycore CPU
NESAP Launched:
transition applications to advanced architectures

2020

NERSC-9: Perlmutter
CPU and GPU nodes
Continued transition of applications and support for complex workflows

2024

NERSC-10

Cori Brings HPC and Data Together

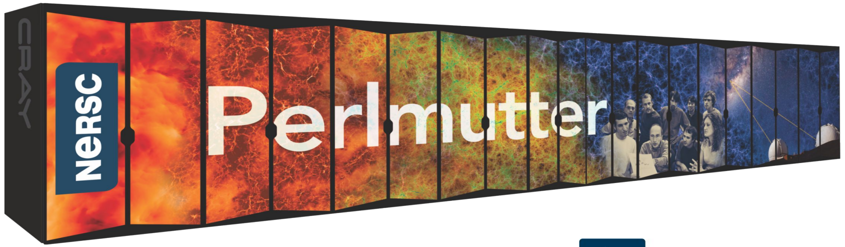
Cori: #13 in Nov 2019 (#5 in Nov 2016) Top 500 list

Gerty Cori: Biochemist and first American woman to win a Nobel Prize in science

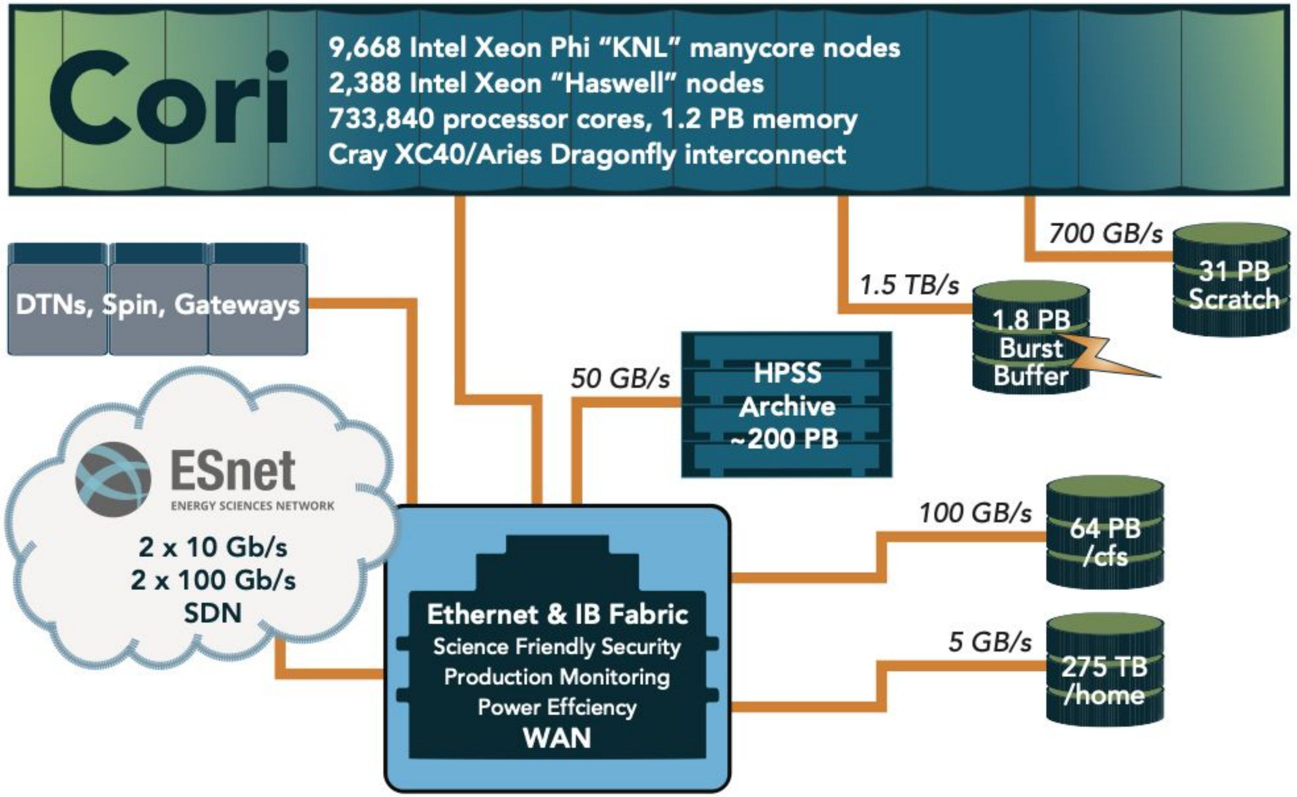
Phase I: 2388 x 32-core Intel Xeon "Haswell" 128 GB DDR4
Also known as "Data Partition" (76,416 cores total)
Phase II: 9688 x 68-core Intel Xeon Phi "KNL" 96 GB DDR4 + 16 GB MCDRAM
(658,784 total cores)

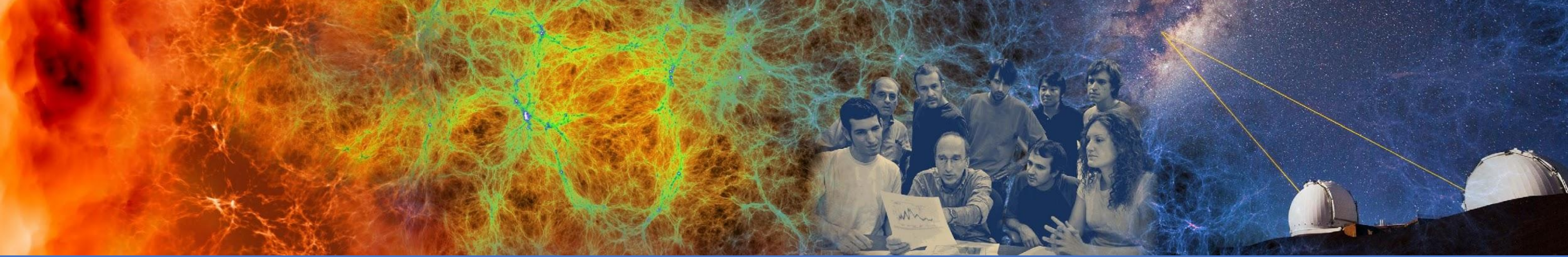
NERSC-9: Perlmutter

- Cray Shasta System providing 3-4x capability of Cori system
  - Phase 1 will arrive in late 2020
- First NERSC system designed to meet needs of both large scale simulation and data analysis from experimental facilities
  - Includes both NVIDIA GPU-accelerated and AMD CPU-only nodes
- Named after Saul Perlmutter: Winner of 2011 Nobel Prize in Physics for discovery of the accelerating expansion of the universe.
  - Works at LBL, is a NERSC user
  - Leader of the Supernova Cosmology Project. Uses supercomputers to combine large-scale simulations with experimental data analysis



NERSC Systems Map 2020





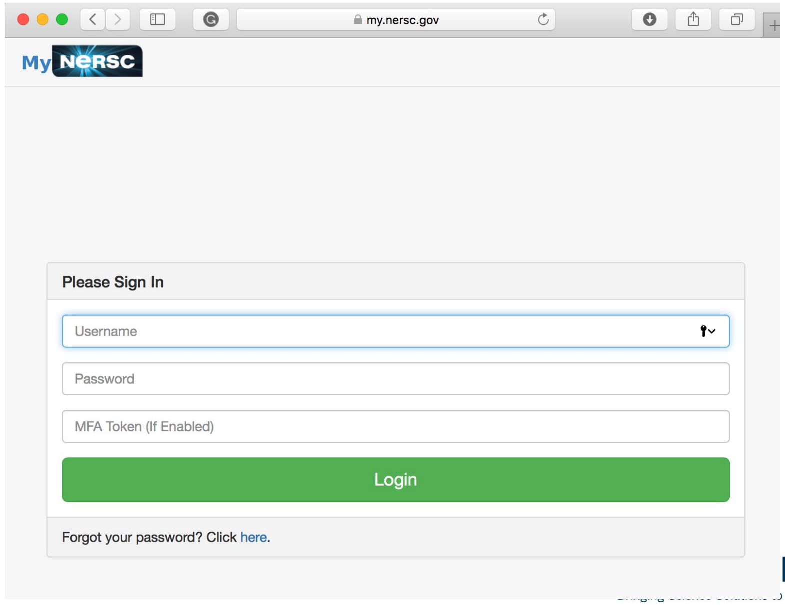
Connecting to NERSC

Multi-Factor Authentication (MFA)

- NERSC password + OTP ("One-Time Password")
  - OTP obtained via the “Google Authenticator” app on your smartphone
  - Alternative/backup option: Authy on desktop https://authy.com/
- MFA is used in login to NERSC systems, web sites, and services
  - Much harder for someone to hack your account
- Mandatory
  - except in special circumstances
- Setup MFA
  - https://docs.nersc.gov/connect/mfa/

MFA Examples

```
<laptop>$ ssh -l elvis cori.nersc.gov
…
Login connection to host cori01 :
Password + OTP:
```



my.nersc.gov

My NERSC

Please Sign In

Username

Password

MFA Token (If Enabled)

Login

Forgot your password? Click here.

Connecting to NERSC: SSH

- All NERSC computational systems are accessible via ssh
- First: you need a terminal program on your desktop/laptop
  - Mac: "terminal" (built-in) or "iTerm2" (https://www.iterm2.com/)
  - Windows: PuTTY (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)
  - Linux: Your own favorite :-)
- If you will use X-forwarding (think GUI) (Note: NX is better!) then you also need an X server
  - Mac: XQuartz (https://www.xquartz.org/)
  - Windows: Cygwin/X (http://x.cygwin.com/)
  - Linux: built in

Example Session (Terminal only)

```
localhost:~elvis> ssh -l elvis cori.nersc.gov

*******************************************************************
*                  NOTICE TO USERS                     *
*                  ---------------                     *
*  Lawrence Berkeley National Laboratory operates this          *
*  computer system under contract to the U.S. Department of     *
*  Energy.  This computer system is the property of the United   *
*  States Government and is for authorized use only.  *Users     *
*  (authorized or unauthorized) have no explicit or implicit    *
*  expectation of privacy.*                            *
*......*                                               *
*******************************************************************

Password:  <enter your SSH password + OTP (one-time-password)  here>
```

Prompt on local system

Notification of acceptable use

Password prompt

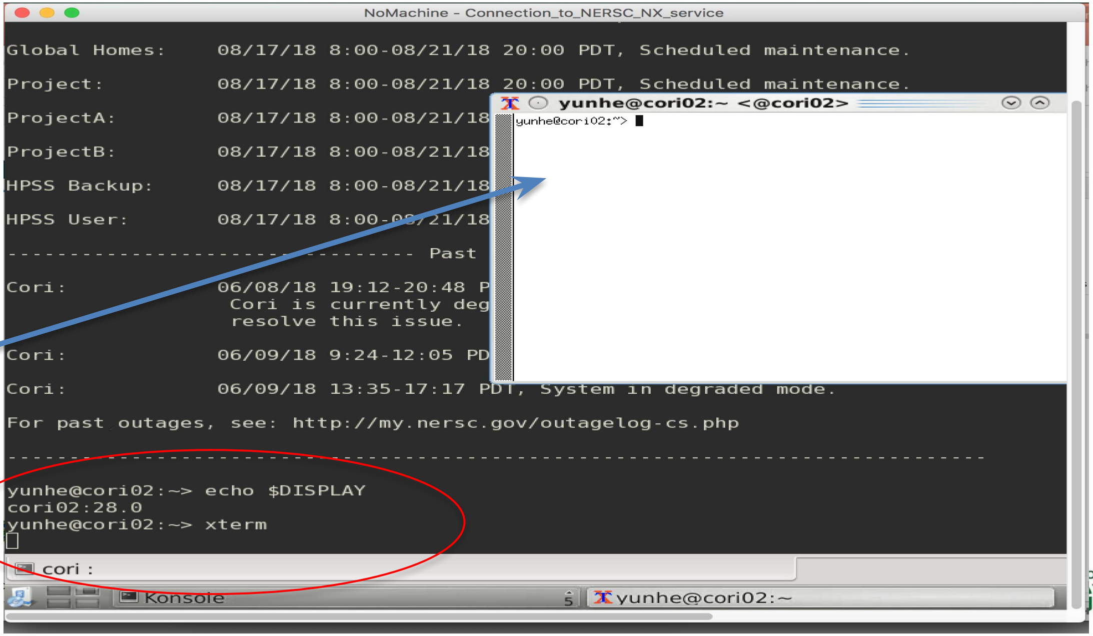
You will login to one of the login nodes (12 on Cori).

To allow X-forwarding to access visualization programs, use the "-Y" flag:

```
localhost% ssh -l elvis -Y cori.nersc.gov
        e/elvis> module load matlab
        e/elvis> matlab
            <MATLAB starts up>
```

Connecting to NERSC: NX (1)

- NERSC recommends using NX instead of SSH X-forwarding since NX is faster and more reliable
- NX is a service for Accelerated X



```
Global Homes:     08/17/18 8:00-08/21/18 20:00 PDT, Scheduled maintenance.
Project:          08/17/18 8:00-08/21/18 20:00 PDT, Scheduled maintenance.
ProjectA:         08/17/18 8:00-08/21/18
ProjectB:         08/17/18 8:00-08/21/18
HPSS Backup:      08/17/18 8:00-08/21/18
HPSS User:        08/17/18 8:00-08/21/18
---------------------------------- Past
Cori:             06/08/18 19:12-20:48 P
                   Cori is currently deg
                   resolve this issue.
Cori:             06/09/18 9:24-12:05 PD
Cori:             06/09/18 13:35-17:17 PDT, System in degraded mode.
For past outages, see: http://my.nersc.gov/outagelog-cs.php
-------------------------------------------------------------------------------
yunhe@cori02:~> echo $DISPLAY
cori02:28.0
yunhe@cori02:~> xterm
```

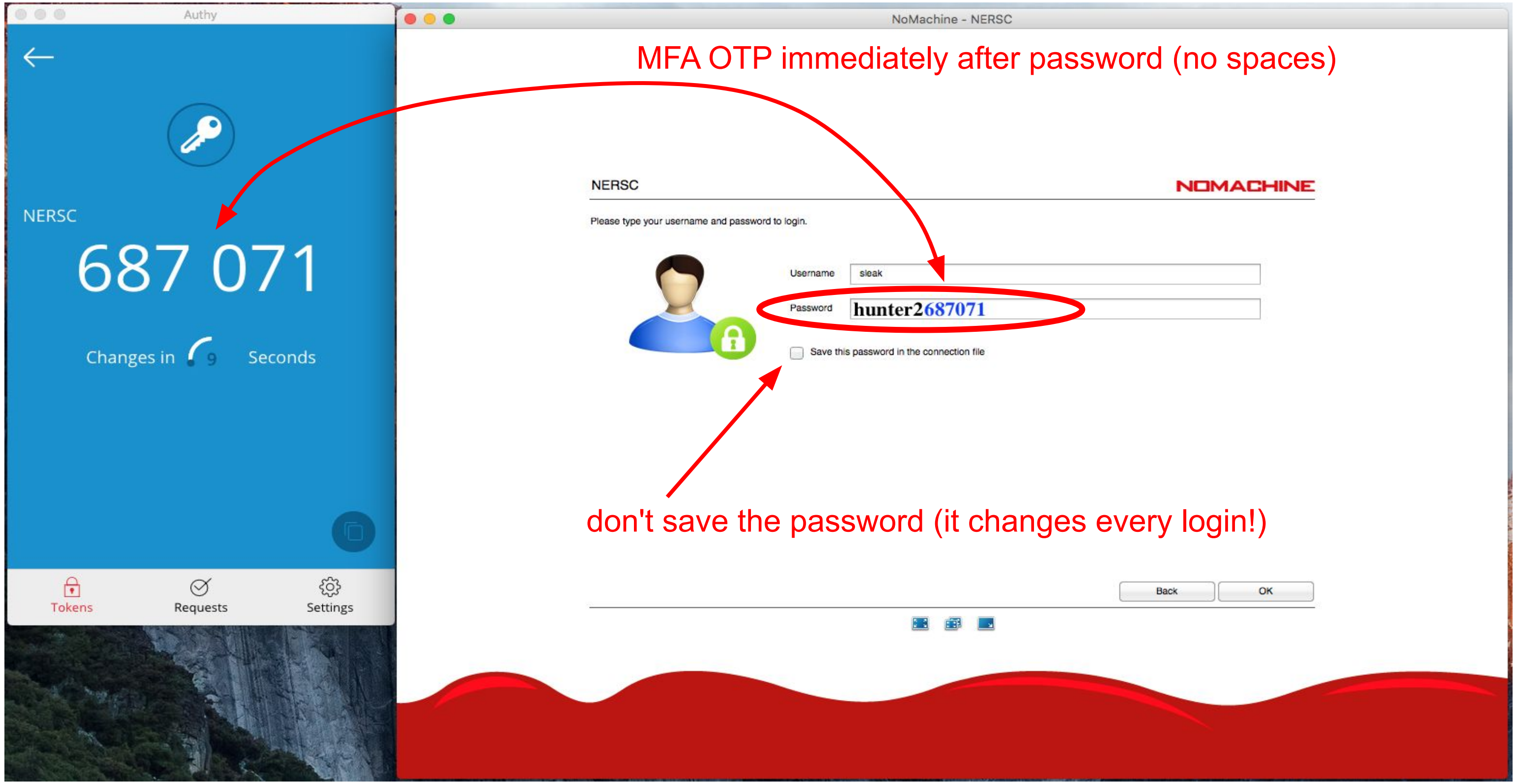
```
yunhe@cori02:~>
```

Opens a new xterm

Connecting to NERSC: NX (2)

- NX also has the benefit of long lasting terminal sessions that can survive between lost internet connections
  - Can reconnect later, even from a different location or computer
- Download and install the Client software: NoMachine
  - Instructions at https://docs.nersc.gov/connect/nx
  - Works on Window/Mac/Linux
- Or use NX Desktop from MyNERSC
  - Temporarily disabled currently
  - Is slower compared to NX Client

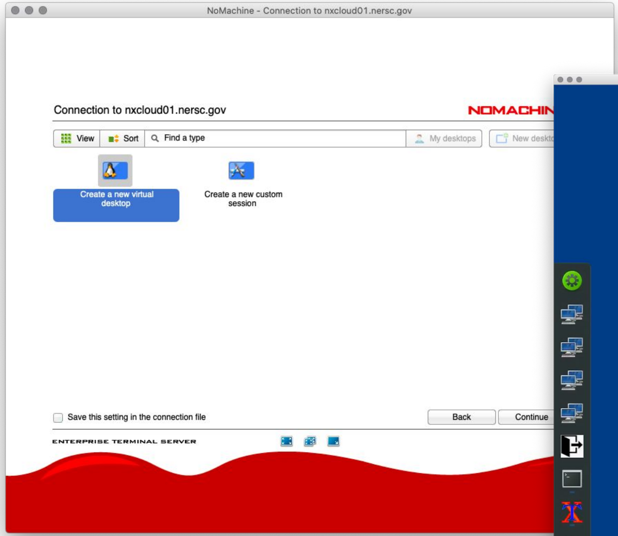
NoMachine Login with MFA

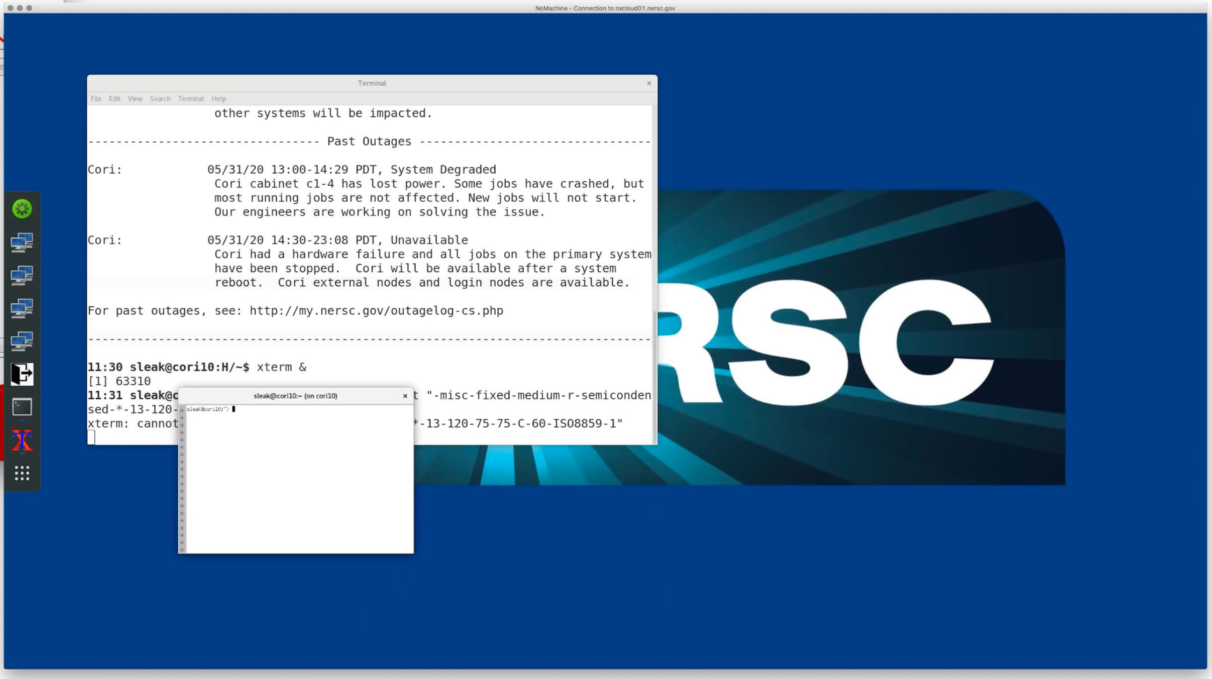


MFA OTP immediately after password (no spaces)

don't save the password (it changes every login!)

NoMachine


NoMachine - Connection to nxcloud01.nersc.gov
Connection to nxcloud01.nersc.gov
NOMACHINE
View
Sort
Find a type
My desktops
New desktop
Create a new virtual desktop
Create a new custom session
Save this setting in the connection file
Back
Continue
ENTERPRISE TERMINAL SERVER


NoMachine - Connection to nxcloud01.nersc.gov
Terminal
File Edit View Search Terminal Help
sleak@cori10:~ (on cori10)

```
                  other systems will be impacted.

------------------------------- Past Outages --------------------------------

Cori:            05/31/20 13:00-14:29 PDT, System Degraded
                  Cori cabinet c1-4 has lost power. Some jobs have crashed, but
                  most running jobs are not affected. New jobs will not start.
                  Our engineers are working on solving the issue.

Cori:            05/31/20 14:30-23:08 PDT, Unavailable
                  Cori had a hardware failure and all jobs on the primary system
                  have been stopped.  Cori will be available after a system
                  reboot.  Cori external nodes and login nodes are available.

For past outages, see: http://my.nersc.gov/outagelog-cs.php

------------------------------------------------------------------------------

11:30 sleak@cori10:H/~$ xterm &
[1] 63310
11:31 sleak@c                                t "-misc-fixed-medium-r-semiconden
sed-*-13-120-                                *-13-120-75-75-C-60-ISO8859-1"
xterm: cannot
```

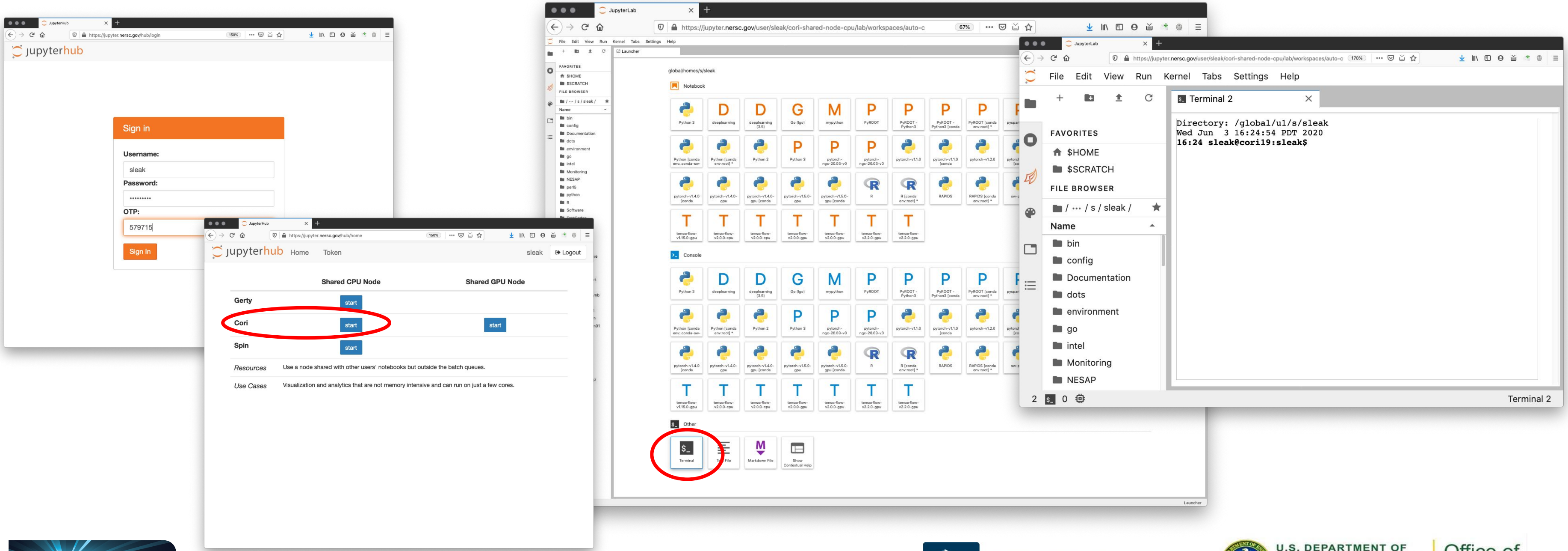
17

sshproxy

- sshproxy.sh creates a short-term (24 hours) certificate
  - Run sshproxy.sh once, then you can ssh to NERSC systems for the next 24 hours before being asked for password+OTP again
- https://docs.nersc.gov/connect/mfa/#sshproxy

Jupyter

You can access Cori from any web browser, via https://jupyter.nersc.gov



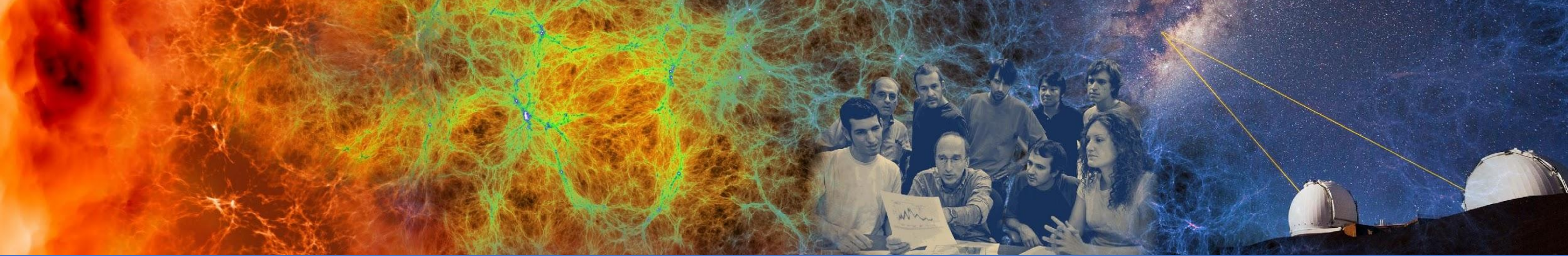
Username: sleak
Password: ••••••••••
OTP: 579715

	Shared CPU Node	Shared GPU Node
Gerty	start	
Cori	start	start
Spin	start	
Resources	Use a node shared with other users' notebooks but outside the batch queues.	
Use Cases	Visualization and analytics that are not memory intensive and can run on just a few cores.	

```
Directory: /global/u1/s/sleak
Wed Jun  3 16:24:54 PDT 2020
16:24 sleak@cori19:sleak$
```

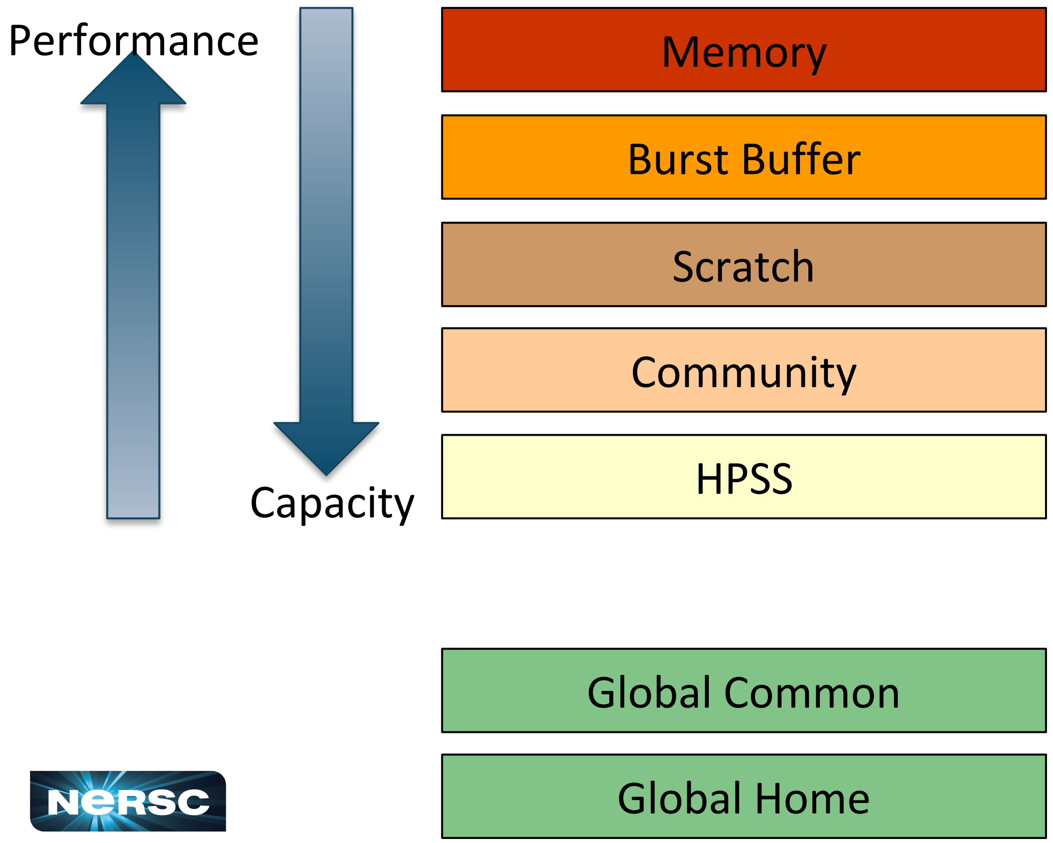
BERKELEY LAB Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY Office of Science



File Systems and Data Management / Transfer

Simplified NERSC File Systems



1.8 PB SSD Burst Buffer on Cori
Cray Datawarp 1.8 TB/s,
temporary for job or campaign

28 PB (Cori) HDD Scratch
Lustre 700 GB/s,
temporary (12 wk purge)

157 PB HDD Community
Spectrum Scale (GPFS)
150 GB/s, permanent

150 PB Tape Archive
HPSS Forever

20 TB SSD Software
Spectrum Scale
Permanent
Faster compiling / Source Code

Global File Systems

Global Home

- Permanent, relatively small storage
- Mounted on all platforms
- NOT tuned to perform well for parallel jobs
- Quota cannot be changed
- Snapshot backups (7-day history)
- **Perfect for storing data such as source code, shell scripts**

Community File System (CFS)

- Permanent, larger storage
- Mounted on all platforms
- Medium performance for parallel jobs
- Quota can be changed
- Snapshot backups (7-day history)
- **Perfect for sharing data within research group**

Local File Systems

Scratch

- Large, temporary storage
- Optimized for read/write operations, NOT storage
- Not backed up
- Purge policy (12 weeks)
- **Perfect for staging data and performing computations**

Burst Buffer

- Temporary storage
  - Can be per job or persistent for multiple users and jobs to access
- High-performance SSD file system
- Available on Cori only
- **Perfect for getting good performance in I/O-constrained codes**

HPSS: Long Term Storage System

- High-Performance Storage System
- Archival storage of infrequently accessed data
- Use hsi and htar to put/get files between NERSC computational systems and HPSS
- More info at
  - https://docs.nersc.gov/filesystems/archive/

DTN: Dedicated Data Transfer System

- Data Transfer Nodes (DTN) are dedicated servers tuned for moving data at NERSC
  - Monitored bandwidth capacity between NERSC & other major facilities such as ORNL, ANL, BNL, SLAC…
  - Can be used to move data internally between NERSC systems and/or NERSC HPSS
- Use NERSC DTNs to move large volumes of data in and out of NERSC or between NERSC systems
- More info at
  - https://docs.nersc.gov/systems/dtn/

Globus Online

- The recommended tool for moving data in&out of NERSC
  - Reliable & easy-to-use web-based service:
    - Automatic retries
    - Email notification of success or failure
  - NERSC managed endpoints for optimized data transfers

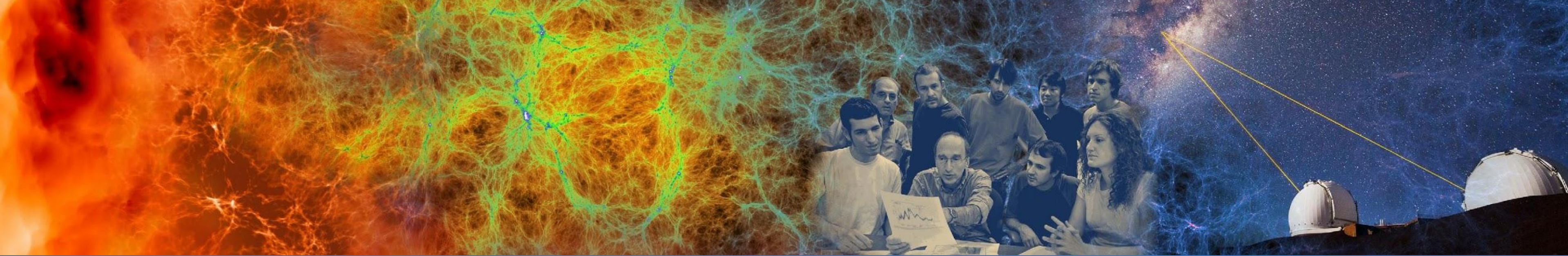
- NERSC documentation

  https://docs.nersc.gov/services/globus/
- Globus extensive documentation

  https://docs.globus.org

Data Transfer General Tips

- Use Globus Online for large, automated or monitored transfers
- scp is fine for smaller, one-time transfers (<100MB)
  - But note that Globus is also fine for small transfers
- Don’t use DTN nodes for non-data transfer purposes
  - Use system login nodes for more general routine tasks
- Don't use your $HOME directory
  - Instead use /global/cfs, $SCRATCH … for better performance
- Plain “cp” is still used for transfers within file systems



Software Environment and Building Applications

Software

- Cray supercomputers OS is a version of Linux
- Compilers are provided on machines
- Libraries: many libraries are provided by vendor, many others provided by NERSC
- Applications: NERSC compiles and supports many software packages (such as chemistry and materials sciences packages) for our users

Modules Environment

- Modules are used to manage the user environment
  - https://docs.nersc.gov/environment/#nersc-modules-environment

`module`	
`list`	To list the modules in your environment

`avail`
 	
`avail -S`
	To list available modules
	To see all available modules: `% module avail`
	To see all available *netcdf* modules: `% module avail -S netcdf`

`load/unload`	To load or unload module
`show/display`	To see what a module loads
`whatis`	Display the module file information
`swap/switch`	To swap two modules For example: to swap architecture target from Haswell to KNL `% module swap craype-haswell craype-mic-knl`
`help`	General help: `$module help` Information about a module: `$ module help PrgEnv-cray`

Default Loaded Modules

Do not do "module purge"

```
yunhe@cori03:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.4                                    13)
gni-headers/5.0.12.0-7.0.1.1_6.27__g3b1768f.ari
  2) nsg/1.2.0                                           14) xpmem/2.2.20-7.0.1.1_4.8__g0475745.ari
  3) altd/2.0                                            15) job/2.2.4-7.0.1.1_3.34__g36b56f4.ari
  4) darshan/3.1.7                                       16) dvs/2.12_2.2.156-7.0.1.1_8.6__g5aab709e
  5) intel/19.0.3.199                                    17) alps/6.6.57-7.0.1.1_5.10__g1b735148.ari
  6) craype-network-aries                                18) rca/2.2.20-7.0.1.1_4.42__g8e3fb5b.ari
  7) craype/2.6.2                                        19) atp/2.1.3
  8) cray-libsci/19.06.1                                 20) PrgEnv-intel/6.0.5
  9) udreg/2.3.2-7.0.1.1_3.29__g8175d3d.ari              21) craype-haswell
 10) ugni/6.0.14.0-7.0.1.1_7.32__ge78e5b0.ari            22) cray-mpich/7.7.10
 11) pmi/5.0.14                                          23) craype-hugepages2M
 12) dmapp/7.1.1-7.0.1.1_4.43__g38cf134.ari
```

5) Compiler   8) Cray Scientific Libraries
20) Programing Environment 21) Target architecture Driver 22) MPI Libraries

Cross-Compile is Needed

- Cori: Haswell compute nodes and KNL compute nodes
- All Cori login nodes are Haswell nodes
- We need to cross-compile
  - Directly compile on KNL compute nodes is very slow
  - Compiles on login nodes; Executables runs on compute nodes
- Binaries built for Haswell can run on KNL nodes, but not vice versa
- Recommends to build separate binaries for each architecture to take advantage of optimizations unique to processor type

Software Environment

- Available compilers: Intel, GNU, Cray
- Use compiler wrappers to build. It calls native compilers for each compiler such as ifort, mpiicc, etc. underneath.
  - Do not use native compilers directly.
  - ftn for Fortran codes: **ftn my_code.F90**
  - cc for C codes: **cc my_code.c**
  - CC for C++ codes: **CC my_code.cc**
- Compiler wrappers add header files and link in MPI and other loaded Cray libraries by default
  - Builds applications dynamically by default. Can add “-static” to build dynamically if chosen

How to Compile for KNL

- The default loaded architecture target module is “craype-haswell” on the Haswell login nodes.
  - This module sets CRAY_CPU_TARGET to haswell
- Best recommendation to build for KNL target
  - module swap craype-haswell craype-mic-knl
  - The above sets CRAY_CPU_TARGET to mic-knl

Building Simple Test Program (1)

- To build on Cori Haswell:
  - Using default Intel compiler:

    ```
    ftn -o mytest mytest_code.F90
    ```

  - Using Cray compiler:

    ```
    module swap PrgEnv-intel PrgEnv-cray
    ftn -o mytest mytest_code.F90
    ```

Building Simple Test Program (2)

- To build on Cori KNL
  - Using default Intel compiler

```
module swap craype-haswell craype-mic-knl
ftn -o mytest mytest_code.F90
```

  - Using Cray compiler

```
module swap PrgEnv-intel PrgEnv-cray
module swap craype-haswell craype-mic-knl
ftn -o mytest mytest_code.F90
```

Compiler Flags

Intel	GNU	Cray	Description/ Comment
-O2	-O0	-O2	default
default, or -O3	-O2 or -O3,-Ofast	default, or -O3	recommended
-qopenmp	-fopenmp	-fopenmp (C/C++) -h omp (Fortran)	OpenMP

Compiler Recommendations

- Will not recommend any specific compiler
  - Intel - better chance of getting processor specific optimizations, especially for KNL
  - Cray compiler – many new features and optimizations, especially with Fortran
  - GNU - widely used by open software
- Try different compilers for potential performance improvement
  - Start with the compilers that vendor/code developers used to minimize the chance of hitting compiler and code bugs

Linking Considerations (1)

- Compiler wrapper will Link with Cray MPI (cray-mpich module is loaded by default), Cray Scientific libraries (cray-libsci module is loaded by default), and most Cray provided libraries and some NERSC provided libraries (need to load corresponding modules) automatically

```
CC parallel_hello.cpp
ftn dgemmx1.f90
module load cray-hdf5
cc h5write.c
```

Linking Considerations (2)

- To link with most NERSC provided libraries, extra include path and libraries need to be added manually, which are usually defined in module files for convenience, such as:

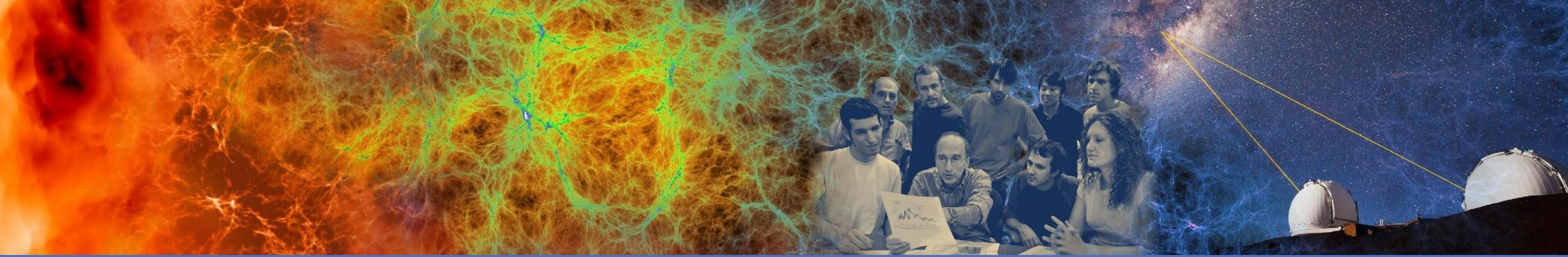
```
module load gsl
ftn test3.f90 $GSL
```

  Use "module show gsl" to see how $GSL is defined

- To link with Intel MKL (Math Kernel Libraries) with Intel compiler, use the "-mkl" flag

```
ftn test1.f90 -mkl
```
  # default to parallel -multi-threaded lib

  The loaded "cray-libsci" will be ignored if -mkl is used



Running Jobs

Jobs at NERSC

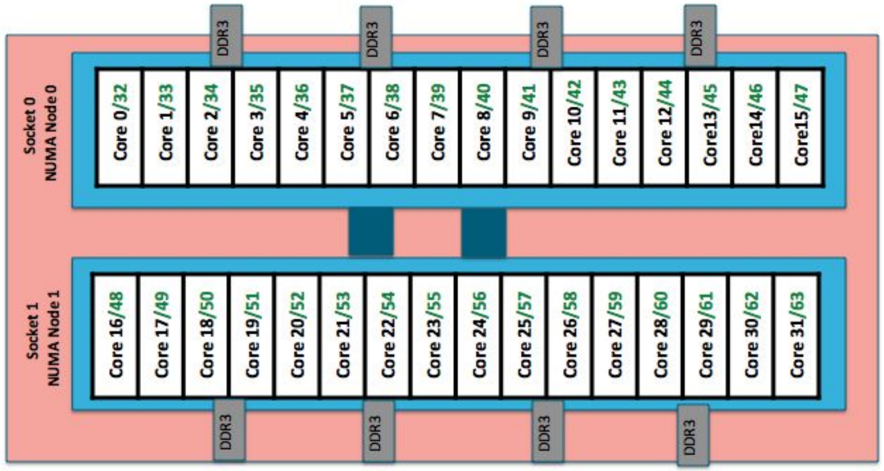
- Most are parallel jobs (10s to 100,000+ cores)
- Also a number of “serial” jobs
    - Typically “pleasantly parallel” simulation or data analysis
- Production runs execute in batch mode
- Our batch scheduler is SLURM
- Debug jobs are supported for up to 30 min
- Batch interactive jobs are supported for up to 4 hrs
- Typical run times are a few to 10s of hours
    - Limits are necessary because of MTBF and the need to accommodate 7,000 users’ jobs

Login Nodes and Compute Nodes

- Login nodes (external)
  - Edit files, compile codes, submit batch jobs, etc.
  - Run short, serial utilities and applications
  - Cori has Haswell login nodes
- Compute nodes
  - Execute your application
  - Dedicated resources for your job
  - Cori has Haswell and KNL compute nodes
  - Binaries built for Haswell can run on KNL nodes, but not vice versa

Cori Haswell Compute Nodes

Cori Phase1 Compute Node



To obtain processor info:

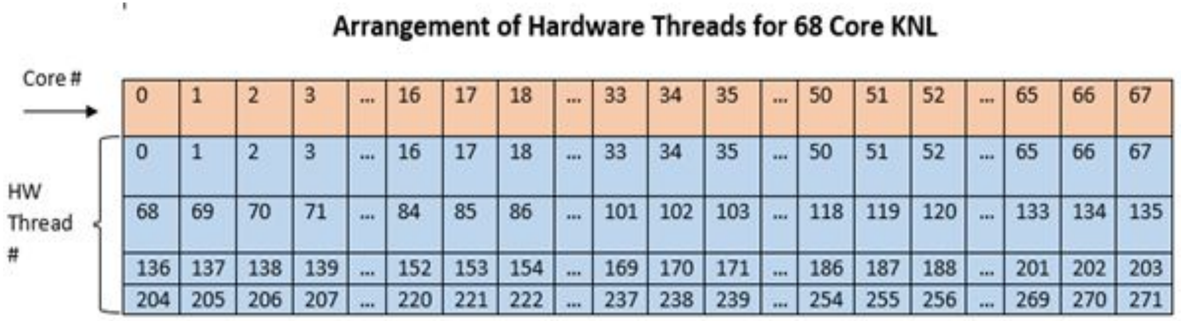
Get on a compute node:
% salloc -N 1 -C …

Then:
% numactl -H
or % cat /proc/cpuinfo
or % hwloc-ls

- Each Cori Haswell node has 2 Intel Xeon 16-core Haswell processors
  - **2 NUMA domains (sockets) per node, 16 cores per NUMA domain. 2 hardware threads per physical core.**
  - **NUMA Domain 0: physical cores 0-15 (and logical cores 32-47)**
    NUMA Domain 1: physical cores 16-31 (and logical cores 48-63)
- Memory bandwidth is non-homogeneous among NUMA domains

Cori KNL Example Compute Nodes

- A Cori KNL node has 68 cores/272 CPUs, 96 GB DDR memory, 16 GB high bandwidth on package memory (MCDRAM)
- Default mode is: quad, cache



Arrangement of Hardware Threads for 68 Core KNL

Core #	0	1	2	3	...	16	17	18	...	33	34	35	...	50	51	52	...	65	66	67
HW Thread #	0	1	2	3	...	16	17	18	...	33	34	35	...	50	51	52	...	65	66	67
	68	69	70	71	...	84	85	86	...	101	102	103	...	118	119	120	...	133	134	135
	136	137	138	139	...	152	153	154	...	169	170	171	...	186	187	188	...	201	202	203
	204	205	206	207	...	220	221	222	...	237	238	239	...	254	255	256	...	269	270	271

- A quad,cache node (default setting) has only 1 NUMA node with all CPUs on the NUMA node 0 (DDR memory). MCDRAM is hidden from the “numactl -H” result since it is a cache.

Submitting Batch Jobs

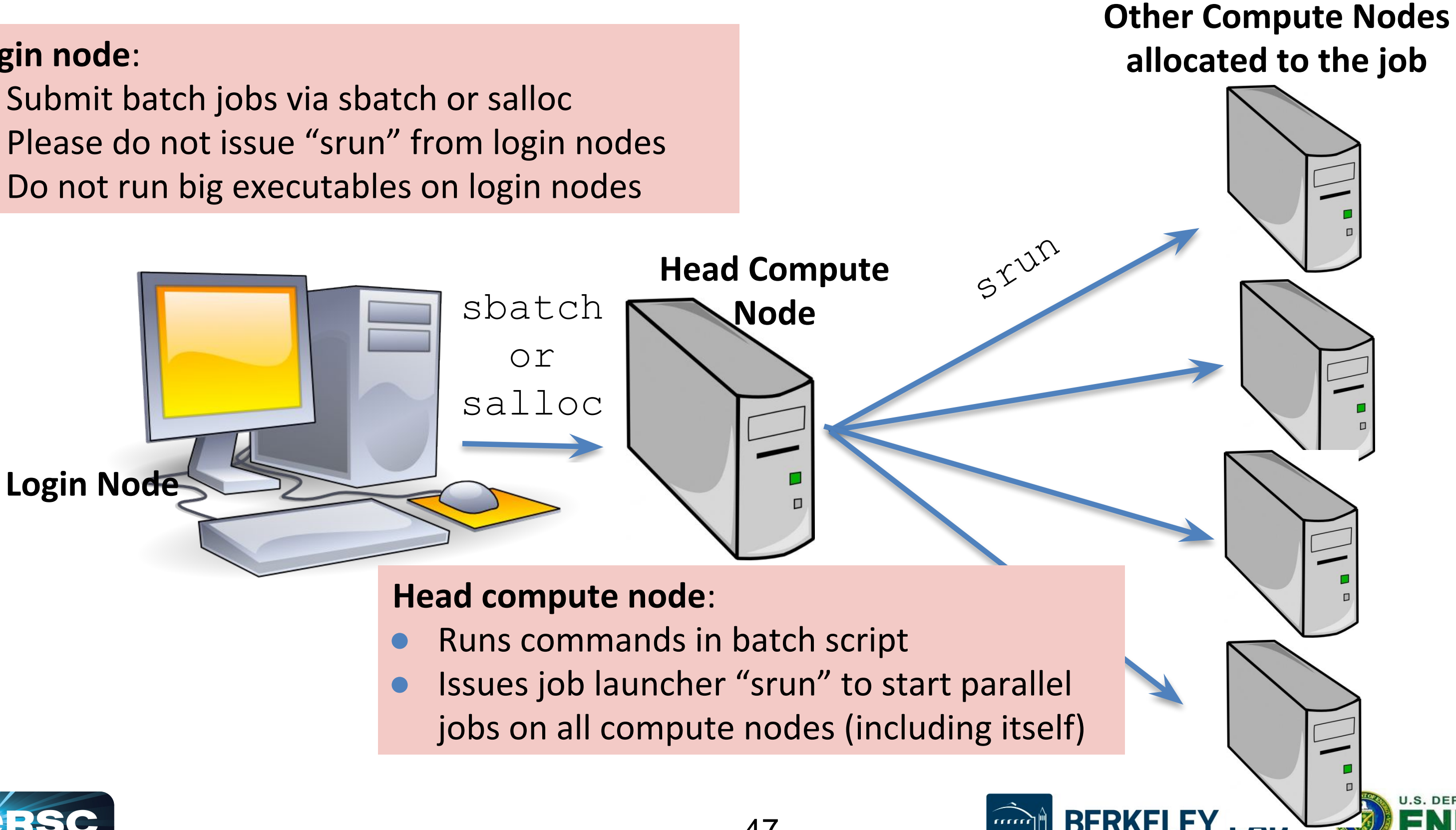
- To run a batch job on the compute nodes you must write a "batch script" that contains:
  - Directives to allow the system to schedule your job
  - An srun command that launches your parallel executable
- A batch job will request resources about which qos, which type of compute nodes, how many nodes, and for how long, etc.
- Submit the job to the queuing system with the sbatch or salloc command

sbatch my_batch_script   or

salloc <command line options>

Launching Parallel Jobs with Slurm

Login node:
- Submit batch jobs via sbatch or salloc
- Please do not issue “srun” from login nodes
- Do not run big executables on login nodes



Head compute node:
- Runs commands in batch script
- Issues job launcher “srun” to start parallel jobs on all compute nodes (including itself)

My First “Hello World” Program

my_batch_script:

```
#!/bin/bash
#SBATCH -q debug
#SBATCH -N 2
#SBATCH -t 10:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob
srun -n 64 ./helloWorld
```

To run via batch queue

```
% sbatch my_batch_script
```

To run via interactive batch

```
% salloc -N 2 -q interactive -C haswell -t 10:00
```

<wait_for_session_prompt. Land on a compute node>

```
% srun -n 64 ./helloWorld
```

Sample Cori Haswell Batch Script

```
#!/bin/bash
#SBATCH --qos=regular
#SBATCH --nodes=4
#SBATCH --time=1:00:00
#SBATCH --constraint=haswell
#SBATCH --license=SCRATCH
#SBATCH --jobname=myjob

srun -n 1280 -c 2 --cpu-bind=cores ./mycode.exe
```

- Need to specify which shell to use for batch script
- Environment is automatically imported

●

Sample Cori Haswell Batch Script

```
#!/bin/bash
#SBATCH --qos=regular
#SBATCH --nodes=4
#SBATCH --time=1:00:00
#SBATCH --constraint=haswell
#SBATCH --license=SCRATCH
#SBATCH --jobname=myjob

srun -n 1280 -c 2 --cpu-bind=cores ./mycode.exe
```

Job directives: instructions for the batch system

- Can use long name or short name (see next slide) to request resources
- Submission QOS (default is "debug")
- How many compute nodes to reserve for your job
- How long to reserve those nodes
- What type of compute nodes to use
- More optional SBATCH keywords

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

srun -n 1280 -c 2 --cpu-bind=cores ./mycode.exe
```

SBATCH optional keywords:

- What file systems my job depends on (prevent to start when there are file system issues)
- What to name my job
- What to name STDOUT files
- What account to charge
- Whether to notify you by email when your job finishes
- …

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

32 MPI tasks per node in this example

- There are 64 logical CPUs (the number Slurm sees) on each node
- “-c” specifies #_logical_CPUs to be allocated to each MPI task
- --cpu-bind is critical especially when nodes are not fully occupied

Sample Cori Haswell Batch Script - Hybrid MPI/OpenMP

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -C haswell

export OMP_NUM_THREADS=8
export OMP_PROC_BIND=true
export OMP_PLACES=threads

srun -n 160 -c 16 --cpu-bind=cores ./mycode.exe
```

4 MPI tasks per node in this example

- Set OMP_NUM_THREADS
- Use OpenMP standard settings for process and thread affinity
- Again, “-c” specifies #_logical_CPUs to be allocated to each MPI task
  - with 4 MPI tasks per node on Haswell, set 64 logical CPUs /4 =16 for ”-c”
  - “-c” value should be >= OMP_NUM_THREADS

Use “shared” QOS to Run Serial Jobs

- The “shared” QOS allows multiple executables from different users to share a node
- Each serial job run on a single physical core of a “shared” node
- Up to 32 (Cori Haswell) jobs from different users depending on their memory requirements

```
#SBATCH -q shared
#SBATCH -t 1:00:00
#SBATCH --mem=4GB
#SBATCH -C haswell
#SBATCH -J my_job
./mycode.x
```

- Do not specify #SBATCH -N”
- Default “#SBATCH -n” is 1
- Default memory is 1,952 MB for Haswell
- Use -n or --mem to request more slots for larger memory
- **Do not use “srun” for serial executable (reduces overhead)**

- Only available on Cori Haswell
- Small parallel job that use less than a full node can also run in the “shared” partition
- https://docs.nersc.gov/jobs/best-practices/#serial-jobs

How to Run Debug and Interactive Jobs

- You can run small parallel jobs interactively on dedicated nodes.
- Debug
  - Max 512 nodes, up to 30 min, run limit 2, submit limit 5

    ```
    % salloc -N 20 -q debug -C haswell -t 30:00
    ```
- Interactive (highly recommend to use this!!)
  - Instant allocation (get nodes in 5 min or reject), run limit 2, submit limit 2
  - Max walltime 4 hrs, up to 64 nodes on Cori (Haswell and KNL combined) per project

    ```
    % salloc -N 2 -q interactive -C knl -t 2:00:00
    ```
  - More information (such as how to find out who in your project is using)
    - https://docs.nersc.gov/jobs/examples/#interactive
    - https://docs.nersc.gov/jobs/interactive/

Advanced Running Jobs Options

- Bundle jobs (multiple “srun”s in one script, sequentially or simultaneously)
- Use Job Arrays to manage collections of similar jobs
- Use job dependency features to chain jobs
- Run variable-time jobs and “flex” qos to run longer jobs
- Use workflow tools to manage jobs
- Use Burst Buffer for faster IO
- Use Shifter for jobs with custom user environment
- Use “xfer” for transferring to/from HPSS
- Use “bigmem” for large memory jobs

Bundle Jobs

Multiple Jobs Sequentially:

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 100
#SBATCH -t 12:00:00
#SBATCH -J my_job
#SBATCH -o my_job.o%j
#SBATCH -L project,SCRATCH
#SBATCH -C haswell

srun -n 3200 ./a.out
srun -n 3200 ./b.out
srun -n 3200 ./c.out
```

- Need to request largest number of nodes needed
- https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-sequentially

Multiple Jobs Simultaneously:

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 9
#SBATCH -t 12:00:00
#SBATCH -J my_job
#SBATCH -o my_job.o%j
#SBATCH -L project
#SBATCH -C haswell

srun -n 44 -N 2 -c2 --cpu-bind=cores ./a.out &
srun -n 108 -N 5 -c2 --cpu-bind=cores ./b.out &
srun -n 40 -N 2 -c2 --cpu-bind=cores ./c.out &
wait
```

- Need to request total number of nodes needed
- No applications are shared on the same nodes
- Make sure to use “&” (otherwise run in sequential) and “wait” (otherwise job exit immediately)
- https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-simultaneously

Job Arrays

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 1:00:00
#SBATCH --array=1-10
#SBATCH -L SCRATCH
#SBATCH -C haswell

cd test_$SLURM_ARRAY_JOB_ID
srun ./mycode.exe
```

- Better managing jobs, not necessary faster turnaround
- Each array task is considered a single job for scheduling
- Use $SLURM_ARRAY_JOB_ID for each individual array task

https://docs.nersc.gov/jobs/examples/#job-arrays

Dependency Jobs

```
cori% sbatch job1
Submitted batch job 1655447

cori06% sbatch --dependency=afterok:5547 job2
or
cori06% sbatch --dependency=afterany:5547 job2
```

https://docs.nersc.gov/jobs/examples/#dependencies

```
cori06% sbatch job1
submitted batch job 1655447

cori06% cat job2
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 1:30:00
#SBATCH -d afterok:1655447
#SBATCH -C haswell
srun -n 16 -c 4 ./a.out

cori06% sbatch job2
```

Use Workflow Management Tools

- These tools can help data-centric science to automate moving data, multi-step processing, and visualization at scales.
- Please do not do below!

```
for i = 1, 10000
        srun -n 1 ./a.out
```

It is inefficient and overwhelms Slurm scheduler

- Available workflow tools include: GNU parallel, Taskfarmer, Fireworks, Nextflow, Papermill, etc.
- One usage case is to pack large number of serial jobs into one script

xfer Jobs

```
#!/bin/bash
#SBATCH -M escori
#SBATCH -q xfer
#SBATCH -t 12:00:00
#SBATCH -J my_transfer

#Archive run01 to HPSS
htar -cvf run01.tar run01
```

- Configured for the purpose of **staging data from HPSS before run or archive result to HPSS after run**
- Avoid wasting NERSC hours if done within large runs
- Runs on external login nodes, via Slurm Server ”escori”.
- Can submit jobs to the xfer QOS from inside another batch script:
    - Add to the end of batch script: “sbatch -M escori -q xfer myarchive.sl”
- https://docs.nersc.gov/jobs/examples/#xfer-queue

bigmem Jobs

```
#!/bin/bash
#SBATCH -M escori
#SBATCH -q bigmem
#SBATCH -N 1
#SBATCH -t 01:00:00
#SBATCH -J my_big_job
#SBATCH -L SCRATCH
#SBATCH --mem=250GB
srun -N 1 -n 1 ./my_big_exe
```

- Runs on external login nodes, via Slurm Server “escori”
- Node is shared among multiple users by default
- Can request exclusive node if needed to run with multiple threads
  - add #SBATCH --exclusive, and use srun -N 1 -c 32 ./my_big_exe
- https://docs.nersc.gov/jobs/examples/#large-memory

Process / Thread / Memory Affinity

- Correct process, thread and memory affinity is the basis for getting optimal performance on Cori Haswell and KNL. It is also essential for guiding further performance optimizations.
  - Process Affinity: bind MPI tasks to CPUs
  - Thread Affinity: bind threads to CPUs allocated to its MPI process
  - Memory Affinity: allocate memory from specific NUMA domains
- Our goal is to promote OpenMP standard settings for portability.
  - OMP_PROC_BIND and OMP_PLACES are preferred to Intel specific KMP_AFFINITY and KMP_PLACE_THREADS settings.
- https://docs.nersc.gov/jobs/affinity/

Can We Just Do a Naive srun?

Example: 16 MPI tasks x 8 OpenMP threads per task on a single 68-core KNL quad,cache node:

```
% export OMP_NUM_THREADS=8
% export OMP_PROC_BIND=spread
% export OMP_PLACES=threads
```

(other choice are "close","master","true","false") — for OMP_PROC_BIND

(other choices are: cores, sockets, and various ways to specify explicit lists, etc.) — for OMP_PLACES

```
% srun -n 16  ./xthi |sort -k4n,6n
 Hello from rank 0, thread 0, on nid02304. (core affinity = 0)
 Hello from rank 0, thread 1, on nid02304. (core affinity = 144)      (on physical core 8)
 Hello from rank 0, thread 2, on nid02304. (core affinity = 17)
 Hello from rank 0, thread 3, on nid02304. (core affinity = 161)      (on physical core 25)
 Hello from rank 0, thread 4, on nid02304. (core affinity = 34)
 Hello from rank 0, thread 5, on nid02304. (core affinity = 178)      (on physical core 42)
 Hello from rank 0, thread 6, on nid02304. (core affinity = 51)
 Hello from rank 0, thread 7, on nid02304. (core affinity = 195)      (on physical core 59)
 Hello from rank 1, thread 0, on nid02304. (core affinity = 0)
 Hello from rank 1, thread 1, on nid02304. (core affinity = 144)
```

It is a mess! thread 0 for rank 0, and thread 1 for rank 1 are on same physical core 0

Importance of -c and --cpu-bind Options

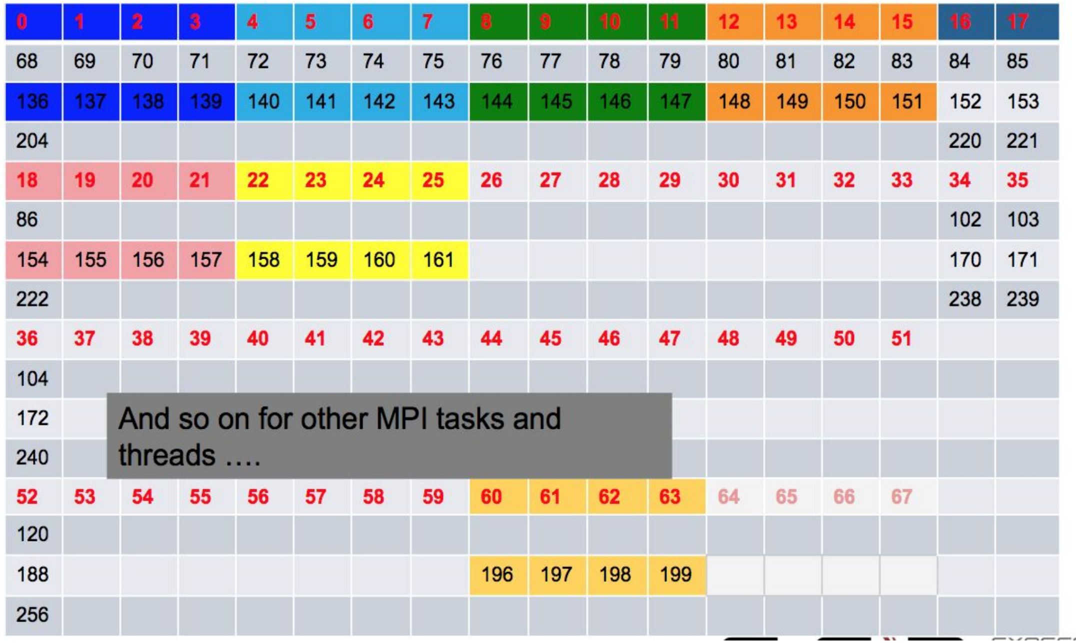
- The reason: 68 cores on KNL is not divisible by #MPI tasks!
  - Each MPI task is getting 68x4/#MPI tasks of logical cores as the domain size
  - MPI tasks are crossing tile boundaries
- Set number of logical cores per MPI task (-c) manually by wasting extra 4 cores on KNL on purpose: 256/#MPI_tasks_per_node.
  - Meaning to use 64 cores only on the 68-core KNL node, and spread the logical cores allocated to each MPI task evenly among these 64 cores.
  - Now it looks good!
  - % srun -n 16 -c 16 --cpu-bind=cores ./xthi

```
Hello from rank 0, thread 0, on nid09244. (core affinity = 0)
Hello from rank 0, thread 1, on nid09244. (core affinity = 136)      (on physical core 0)
Hello from rank 0, thread 2, on nid09244. (core affinity = 1)
Hello from rank 0, thread 3, on nid09244. (core affinity = 137)      (on physical core 1)
```

Now It Looks Good!

🟦	MPI rank 0
🟦	MPI rank 1
🟩	MPI rank 2
🟧	MPI rank 3
🟥	MPI rank 4
🟨	MPI rank 5
….	
🟨	MPI rank 15

Process/thread affinity are good! (Marked first 6 and last MPI tasks only)



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
204																220	221
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
86																102	103
154	155	156	157	158	159	160	161									170	171
222																238	239
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51		
104																	
172																	
240																	
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67		
120																	
188								196	197	198	199						
256																	

And so on for other MPI tasks and threads ….

Sample Job Script to Run on KNL Nodes

Sample Job script (MPI+OpenMP)

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -q regular
#SBATCH -t 1:00:00
#SBATCH -L SCRATCH
#SBATCH -C knl,quad,cache

export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=4
srun -n 128 -c 4 --cpu_bind=cores ./a.out
```

With the above two OpenMP envs, each thread is now pinned to a single CPU within each core

Process and thread affinity

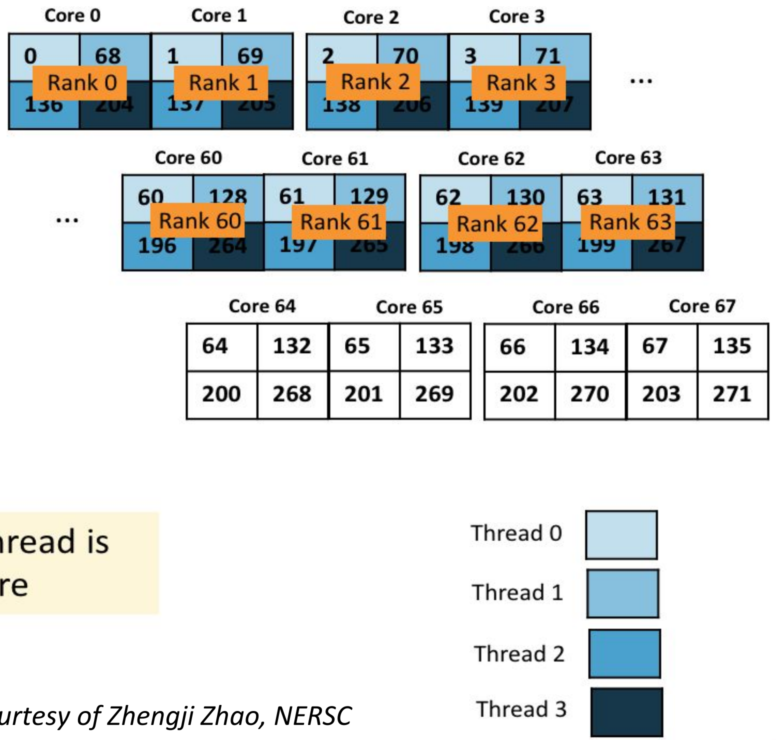
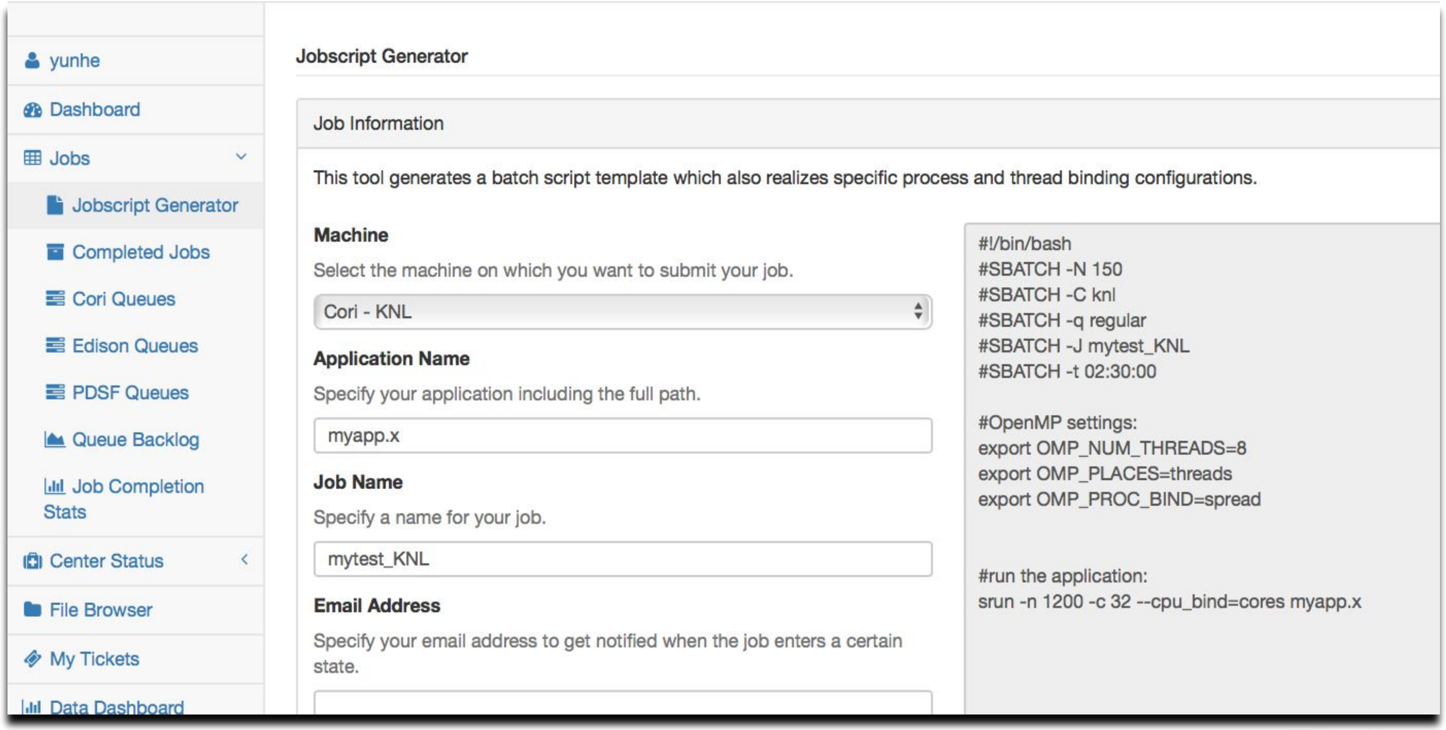


Illustration Courtesy of Zhengji Zhao, NERSC

NERSC Job Script Generator

https://my.nersc.gov/script_generator.php



Jobscript Generator

Job Information

This tool generates a batch script template which also realizes specific process and thread binding configurations.

Machine

Select the machine on which you want to submit your job.

Cori - KNL

Application Name

Specify your application including the full path.

myapp.x

Job Name

Specify a name for your job.

mytest_KNL

Email Address

Specify your email address to get notified when the job enters a certain state.

```
#!/bin/bash
#SBATCH -N 150
#SBATCH -C knl
#SBATCH -q regular
#SBATCH -J mytest_KNL
#SBATCH -t 02:30:00

#OpenMP settings:
export OMP_NUM_THREADS=8
export OMP_PLACES=threads
export OMP_PROC_BIND=spread


#run the application:
srun -n 1200 -c 32 --cpu_bind=cores myapp.x
```

Monitoring Your Jobs

- Once your job is submitted, it enters the queue and will start when resources are available
- Overall job priorities are a combination of QOS, queue wait time, job size, wall time request (and fair share).
- You can monitor with
  - `squeue`
  - `sqs`
  - `sacct`
- On the web
  - https://my.nersc.gov
    - Cori Queues, Queue backlogs, Queue Wait Times (statistics data)
  - https://www.nersc.gov/users/live-status/ □ Queue Look
  - https://iris.nersc.gov the "Jobs" tab

squeue: Slurm Batch Queue Display

```
yunhe@cori05:~> yunhe@cori09:~> squeue -a |more
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          31593007 regular_k allHSQf2    detar CG    5:46:29     13 nid[02568-02569,03678,03816,03888-03889,0726
5,07806,07811,09911-09912,10697,10806]
          31611508    shared run_each  cemitch CG       3:12      1 nid00553
          31611509    shared run_each  cemitch CG       3:12      1 nid00552
          31146718 regular_k hello_up bonachea PD       0:00      1 (ReqNodeNotAvail, UnavailableNodes:nid[02655
,02994,03002,03446,03465,03818,03912,04028-04029,04202,04219,04408,04466,04950,05087,05152,05163,05444,05689,060
96-06099,06580,06662,06902,06948,07462,07813,08029,08215,08251,08562,08603,08815,09133,09408-09419,09424-09487,0
9492-09547,09552-09599,09762,11062,11247,11557,11835,11905])
          31612924  genepool align-70  qc_user PD       0:00      1 (Resources)
          31612927  genepool filter-7  qc_user PD       0:00      1 (Priority)
          31612929  genepool align-70  qc_user PD       0:00      1 (Priority)
          31611879 debug_knl benchmar   junmin PD       0:00      8 (Dependency)
          31611883 debug_knl benchmar   junmin PD       0:00    128 (Dependency)
          31611888 debug_knl benchmar   junmin PD       0:00     16 (Dependency)
          31611897 debug_hsw     test startsev PD       0:00     32 (Dependency)
          31611902 debug_knl benchmar   junmin PD       0:00     32 (Dependency)
    31612757_[3-5] debug_hsw runme.sh  kkrizka PD       0:00      1 (QOSMaxJobsPerUserLimit)
...
```

- By default, “squeue” displays all users jobs.
- Use “squeue -u" to display your own jobs.
- See “squeue --help” or “man squeue” for more details.

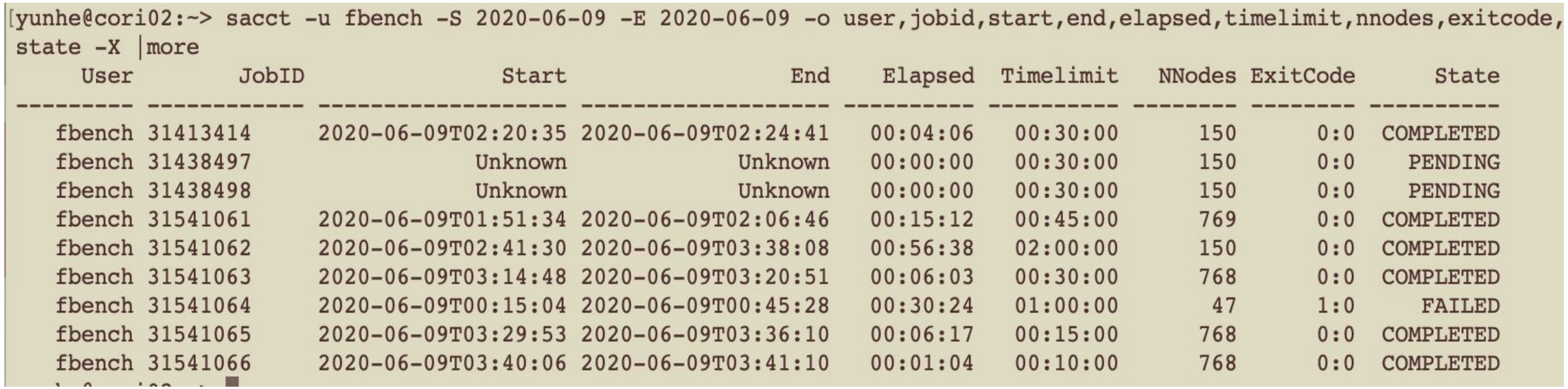
sqs: NERSC Custom Batch Queue Display

```
yunhe@cori05:~> sqs
JOBID       ST  USER   NAME        NODES  REQUESTED   USED  SUBMIT               QOS        SCHEDULED_START      FEATURES         REASON
110901xx    PD  fxxxx  mxxx        1536      5:00     0:00  2018-03-20T10:49:23  regular_0  2018-03-22T06:30:00  haswell          Resources
110901xx    PD  fxxxx  run.xxx*    1537     20:00     0:00  2018-03-20T10:51:03  regular_0  2018-03-22T06:30:00  haswell          Resources
110823xx    PD  fxxxx  gxxx         300     30:00     0:00  2018-03-19T23:05:24  regular_1  avail_in_~1.6_days   haswell          Priority
110823xx    PD  fxxxx  run-xx       768     20:00     0:00  2018-03-19T23:05:33  regular_1  avail_in_~1.6_days   haswell          Priority
110823xx    PD  fxxxx  rxxxx       1536     20:00     0:00  2018-03-19T23:05:04  regular_0  N/A                  haswell          JobHeldUser
110823xx    PD  fxxxx  axxxxxxxx*  1536     30:00     0:00  2018-03-19T23:05:16  regular_0  N/A                  haswell          JobHeldUser
111152xx    PD  fxxxx  run.xxx      769   2:00:00     0:00  2018-03-21T09:39:29  regular_1  avail_in_~3.0_days   knl&quad&cache   None
<omitted…>
```

```
yunhe@cori05:~> sqs2
JOBID           ST USER      NAME          NODES TIME_LIMIT       TIME  SUBMIT_TIME          QOS             START_TIME           FEATURES
NODELIST(REASON)
31567887        PD fxxx      wrxx          512        15:00       0:00  2020-06-09T23:11:27  debug_knl       2020-06-10T00:56:00  knl&quad&cache (Resources)
31438456        PD fxxx      mpixxx        150        30:00       0:00  2020-06-07T12:42:04  regular_1       N/A                  haswell        (Resources)
31543103        PD fxxx      mpixxx        3          30:00       0:00  2020-06-09T00:22:12  regular_1       N/A                  haswell        (Priority)
31402334        R  fxxx      Nxxxxx        1       12:00:00    4:27:45  2020-06-05T23:59:19  regular_1       2020-06-09T19:28:54  knl&quad&cache nid10273
<omitted….>
```

- By default, “sqs” displays your own jobs. Use “sqs -a" to display all users jobs.
- See “sqs --help” for more details.
- sqs2 is a simplified NERSC wrapper for the Slurm "squeue" command with a chosen default format. It takes all allowed flags in “squeue”.
- “sqs2” will be renamed to “sqs” in July.

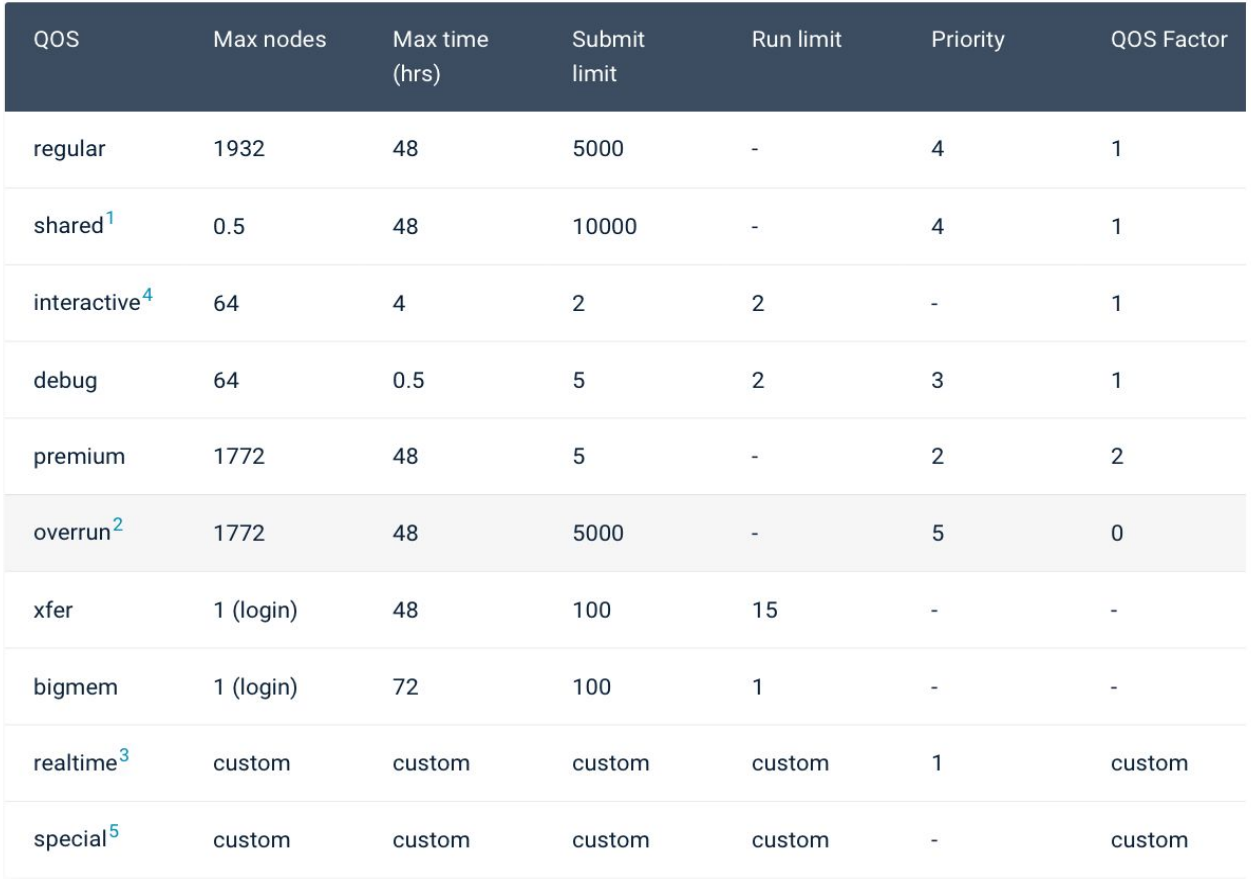
sacct: Query Completed and Pending Jobs



```
[yunhe@cori02:~> sacct -u fbench -S 2020-06-09 -E 2020-06-09 -o user,jobid,start,end,elapsed,timelimit,nnodes,exitcode,
 state -X |more
     User        JobID               Start                 End    Elapsed  Timelimit   NNodes ExitCode      State
--------- ------------ ------------------- ------------------- ---------- ---------- -------- -------- ----------
   fbench 31413414     2020-06-09T02:20:35 2020-06-09T02:24:41   00:04:06   00:30:00      150      0:0  COMPLETED
   fbench 31438497                 Unknown             Unknown   00:00:00   00:30:00      150      0:0    PENDING
   fbench 31438498                 Unknown             Unknown   00:00:00   00:30:00      150      0:0    PENDING
   fbench 31541061     2020-06-09T01:51:34 2020-06-09T02:06:46   00:15:12   00:45:00      769      0:0  COMPLETED
   fbench 31541062     2020-06-09T02:41:30 2020-06-09T03:38:08   00:56:38   02:00:00      150      0:0  COMPLETED
   fbench 31541063     2020-06-09T03:14:48 2020-06-09T03:20:51   00:06:03   00:30:00      768      0:0  COMPLETED
   fbench 31541064     2020-06-09T00:15:04 2020-06-09T00:45:28   00:30:24   01:00:00       47      1:0     FAILED
   fbench 31541065     2020-06-09T03:29:53 2020-06-09T03:36:10   00:06:17   00:15:00      768      0:0  COMPLETED
   fbench 31541066     2020-06-09T03:40:06 2020-06-09T03:41:10   00:01:04   00:10:00      768      0:0  COMPLETED
```

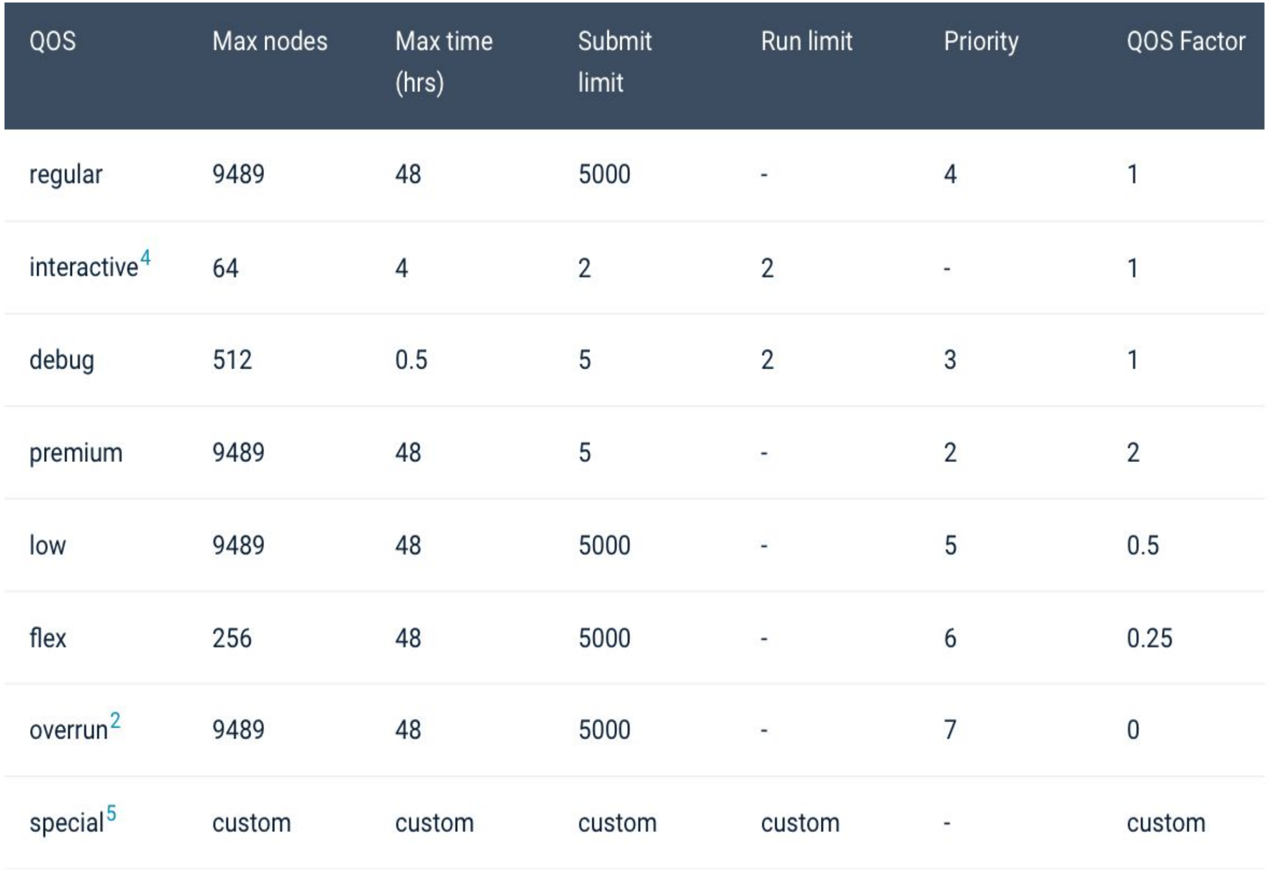
- Maximum query duration is one month (subject to change)
- Detailed job steps info will be displayed without “-X” flag
- Many more job fields can be queried. See “sacct --help” or “man sacct” for more details.

Cori Haswell Queue Policy (as of June 2020)



QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	QOS Factor
regular	1932	48	5000	-	4	1
shared[1]	0.5	48	10000	-	4	1
interactive[4]	64	4	2	2	-	1
debug	64	0.5	5	2	3	1
premium	1772	48	5	-	2	2
overrun[2]	1772	48	5000	-	5	0
xfer	1 (login)	48	100	15	-	-
bigmem	1 (login)	72	100	1	-	-
realtime[3]	custom	custom	custom	custom	1	custom
special[5]	custom	custom	custom	custom	-	custom

Cori KNL Queue Policy (as of June 2020)



QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	QOS Factor
regular	9489	48	5000	-	4	1
interactive[4]	64	4	2	2	-	1
debug	512	0.5	5	2	3	1
premium	9489	48	5	-	2	2
low	9489	48	5000	-	5	0.5
flex	256	48	5000	-	6	0.25
overrun[2]	9489	48	5000	-	7	0
special[5]	custom	custom	custom	custom	-	custom

How Your Jobs are Charged (1)

- Unit: NERSC Hours
- Each architecture has a base charge per node hour used:
  - Cori Haswell: 140
  - Cori KNL: 80
- Modification to base charge by QOS used:
  - premium: 2.0
  - regular: 1.0 (default)
  - low: 0.5
  - flex: 0.25
  - overrun: 0
  - shared: fraction of the node used
- On Cori KNL
  - Jobs requesting 1024 or more nodes get a 20% discount

How your Jobs are Charged (2)

- Your project is charged for each node your job was allocated for the entire duration of your job.
  - The minimum allocatable unit is a node (*except for the “shared” QOS*).
  - Example: 4 Cori Haswell nodes, run for 1 hour with “premium” QOS
    NERSC hours = 4 * 1 hour * 140 * 2 = 1120
  - “shared” jobs are charged with # of physical cores used instead of the entire node.
- If you have access to multiple projects, pick which one to charge in your batch script

```
#SBATCH -A project_name
```

How are Jobs Scheduled

- Each job has its priority value, composed of qos, job age, and a small value of fairshare.
- There are two Slurm schedulers: main and backfill.
- Every few minutes, the main scheduler schedules jobs in the order of the priority list a few days into the future.
  - Jobs are only eligible to be scheduled if they've reached a priority threshold.
  - Currently only 2 jobs per qos per user are considered for scheduling.
- The backfill scheduler then schedules small and short jobs to run if they will not affect the start time of those jobs that are already scheduled by the main scheduler.

Tips for Getting Better Throughput

- Line jumping is allowed, but it may cost more (with “premium” QOS)
- Submit shorter jobs, they are easier to schedule
  - Checkpoint to break up long jobs, use variable time
  - Short jobs can take advantage of ‘backfill’ opportunities
  - Run short jobs just before maintenance
  - Run variable-time jobs; use “flex” QOS
- Make sure the wall clock time you request is accurate
  - Larger shorter jobs are easier to schedule than long smaller jobs
  - Many users unnecessarily request the largest wall clock time possible as default
- Check queue backlogs and queue wait times
  - https://my.nersc.gov/backlog.php
  - https://my.nersc.gov/queuewaittimes.php

Large Jobs Considerations

- sbcast your executables to compute nodes before srun:

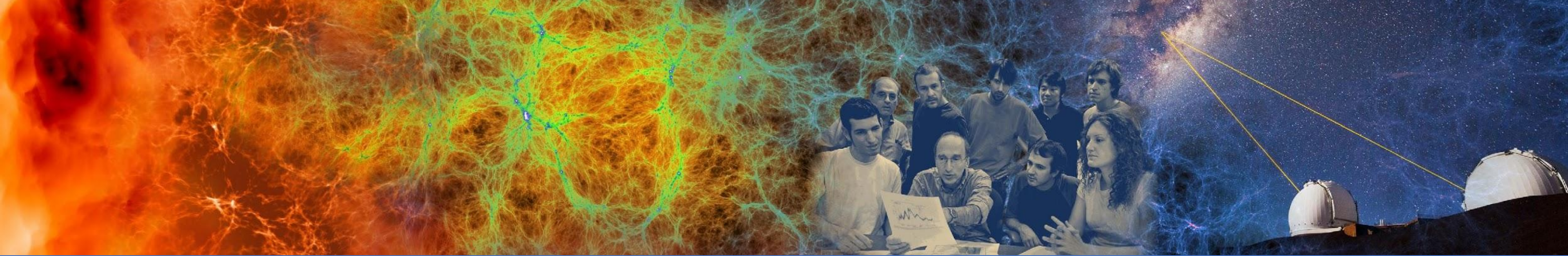
```
    sbcast --compress=lz4 /path/to/exe /tmp/exe
    srun /tmp/exe
```

  https://docs.nersc.gov/jobs/best-practices/#large-jobs

- Consider to build statically to run large jobs.
  - There may be considerable startup delays for running large jobs of dynamic executables.
- Consider to use shifter for large jobs using shared libraries.
- Consider to use burst buffer for jobs doing large IO.

Other Running Jobs Considerations

- Remember to compile separately for each type of compute nodes
- Running jobs from global homes is strongly discouraged
  - IO is not optimized
  - The global homes file system access on compute nodes is much slower than from $SCRATCH
  - It may also cause negative impact for other users interactive response on the system
- Consider to put your project's shared software in /global/common/software/<project>
  - It is mounted read-only on compute nodes, so has less impact than other GPFS file systems (global homes or community file system)
- Consider to adopt workflow tools for better managing your jobs



Data Analytics Software and Services

Cori's Data Friendly Features

12 32-core Haswell
500 GB Login Nodes

Jupyter Notebook Node

768 GB "bigmem" Haswell Nodes

Pipeline/Workflow Management Nodes

Serial Queue
Shared-node Queue
Transfer Queue

Containerized Environments

Real-Time Queues for Co-Scheduling w/Experiments

Streaming Data to Compute Nodes

External Network Access to/from Compute Nodes

Cray DataWarp: Burst Buffer for I/O acceleration

Interactive Queue:
64 Nodes x 4 Hours

Production Data Software Stack

Capabilities	Technologies
Data Transfer + Access	globus online, GridFTP, jupyter, !m, python django, newt
Workflows	FireWorks, TaskFarmer
Data Management	HDF, netCDF, ROOT, mongoDB, MySQL, PostgreSQL
Data Analytics	python, Spark, TensorFlow, K, R, julia, Caffe, PYTORCH
Data Visualization	visit, ParaView

Data Analytic Software Services

- Science Gateways
- Databases
- Shifter
- Burst Buffer
- Python
- Jupyter
- Machine Learning / Deep Learning
- Workflows
- And more …

Access for External Collaborators

- Science Gateways (web portals)
  - NERSC supports project-level public http access
    - Project specific area can be created:

      /global/cfs/cdirs/<your_project>/www
    - These are available for public access under the URL:

      http://portal.nersc.gov/cfs/<your_project>
  - Each repo has a /project space, can publish as above
  - Special Science Gateways can be created. Sophisticated ones can be made with SPIN: https://docs.nersc.gov/services/spin/getting_started/
  - Details at: https://docs.nersc.gov/services/science-gateways/
- FTP Upload Service (external user to share data with NERSC user) https://www.nersc.gov/users/job-logs-statistics/storage-and-file-systems/nersc-ftp-upload-service/

Databases

- Relational / SQL Databases
  - MySQL and PostgreSQL, good for:

      structured data (have a 'Schema')

      Relational (tables of rows and columns)

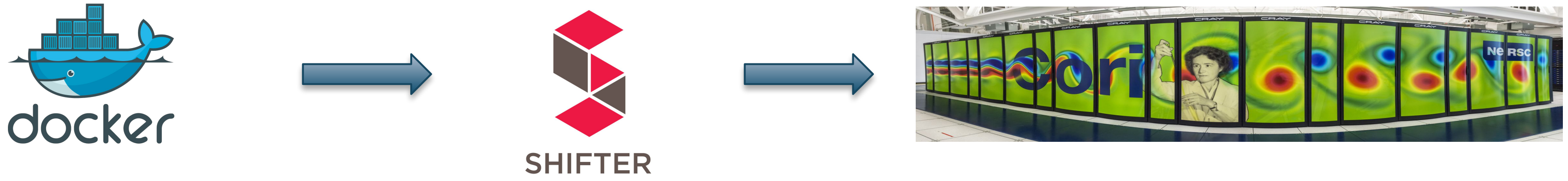
      Mid-Size, <= several GB in total
- NoSQL / Schema-less Databases
  - MongoDB, good for:

      Un-Structured Data ('Schema-less')

      Mid-Size to Large, e.g. 10 GB of Text
- More info and how to request a database: https://docs.nersc.gov/services/databases/

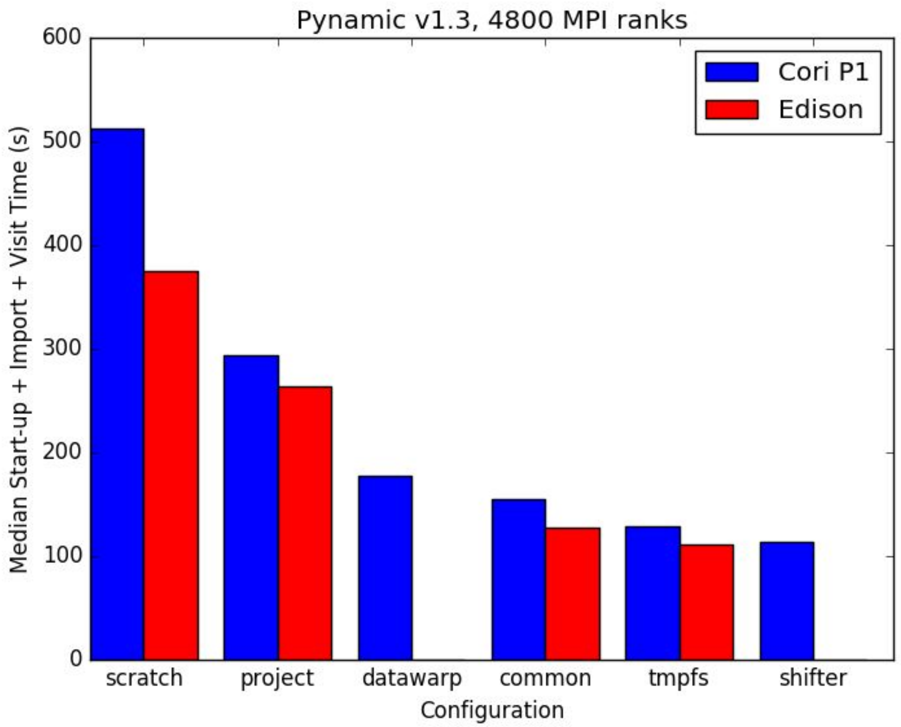
Shifter

- NERSC R&D effort, in collaboration with Cray, to support Docker Application images
- “Docker-like” functionality on the Cray and HPC Linux clusters. Enables users to run custom environments on HPC systems.
- Addresses security issues in a robust way
- Efficient job-start & Native application performance



https://docs.nersc.gov/development/shifter/how-to-use/

Shifter Accelerates Python Applications



Create an Image with Docker

Dockerfile

```
FROM ubuntu:14.04
MAINTAINER Shane Canon scanon@lbl.gov
# Update packages and install dependencies
RUN apt-update -y && \
   apt-get install -y build-essential

# Copy in the application
ADD . /myapp
# Build it
RUN cd /myapp && \
    make && make install
```

```
laptop> docker build -t scanon/myapp:1.1 .
laptop> docker push scanon/myapp:1.1
```

Use the Image with Shifter

```
#!/bin/bash
#SBATCH -N 16 -t 20
#SBATCH --image=scanon/myapp:1.1

module load shifter
export TMPDIR=/mnt
srun -n 16 shifter /myapp/app
```

Submit script
job.sl

```
cori> shifterimg pull scanon/myapp:1.1
cori> sbatch ./job.sl
```

Shifter and MPI

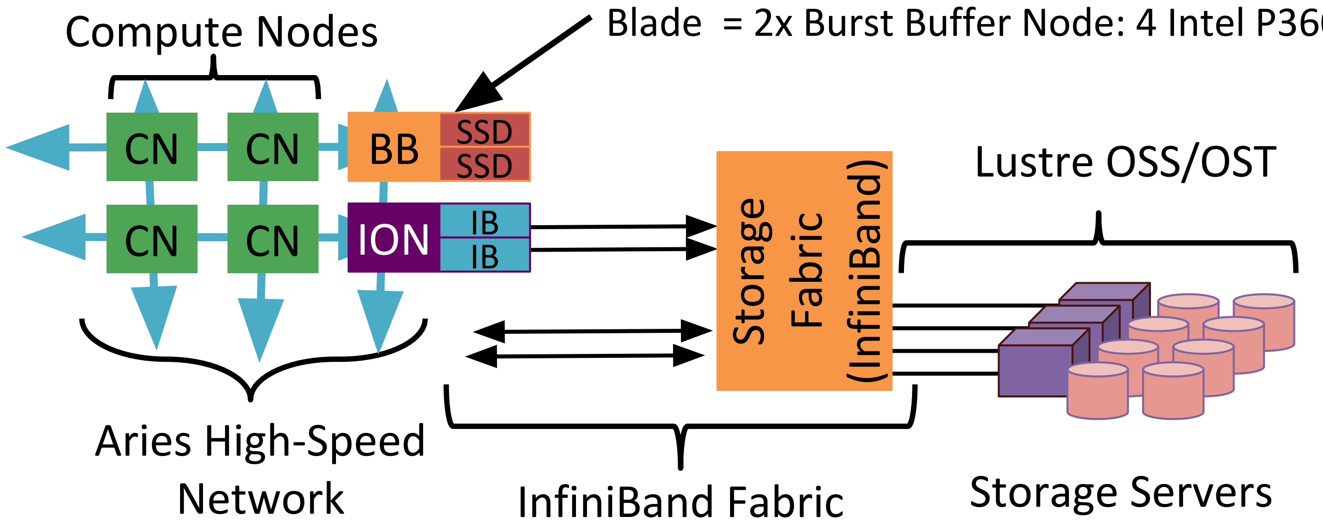
```
# This example makes use of an Ubuntu-based NERSC base image
# that already has MPI built and installed.
# Shifter automatically maps in appropriate libraries at run time.

FROM nersc/ubuntu-mpi:14.04
ADD helloworld.c /app/
RUN cd /app && mpicc helloworld.c -o /app/hello
ENV PATH=/usr/bin:/bin:/app:/usr/local/bin
```

```
cori> shifterimg pull scanon/myapp:1.1
cori> salloc -n 128 --image=scanon/myapp:1.1 -C haswell
% srun -n 128 shifter /myapp/app
```

Use Burst Buffer for Faster IO

- Cori has 1.8PB of SSD-based “Burst Buffer” to support I/O intensive workloads
- Jobs can request a job-temporary BB filesystem, or a persistent (up to a few weeks) reservation
- More info
  - https://docs.nersc.gov/jobs/examples/#burst-buffer

Burst Buffer Architecture



	Burst Buffer	Lustre
Nodes	288	248
Capacity (PB)	1.8	28

- DataWarp software (integrated with SLURM WLM) allocates portions of available storage to users per-job (or ‘persistent’).
- Users see a POSIX filesystem
- Filesystem can be striped across multiple BB nodes (depending on allocation size requested)

Burst Buffer Example

```
#!/bin/bash
#SBATCH -q regular -N 10 -C haswell -t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=$SCRATCH/inputs destination=$DW_JOB_STRIPED/inputs \ type=directory
#DW stage_in source=$SCRATCH/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs destination=/lustre/outputs \  type=directory
srun my.x --indir=$DW_JOB_STRIPED/inputs --infile=$DW_JOB_STRIPED/file.dat \
--outdir=$DW_JOB_STRIPED/outputs
```

- ‘type=scratch’ – duration just for compute job (i.e. not ‘persistent’)
- ‘access_mode=striped’ – visible to all compute nodes and striped across multiple BB nodes
- Data ‘stage_in’ before job start and ‘stage_out’ after

Python

- Extremely popular interpreted language, continuing to grow
- Libraries like NumPy, SciPy, scikit-learn commonly used for scientific analysis
- Are used for ML/DL
- NERSC Python is Anaconda
- https://docs.nersc.gov/programming/high-level-environments/python/
- Do not use /usr/bin/python, instead:

  module load python

  which already includes basic packages: numpy, scipy, mpi4py

Your Own Python Conda Environment

- To make a custom env

```
module load python
conda create -n myenv python=3.7
source activate myenv
conda (or pip) install your_custom_package
###import antigravity
source deactivate myenv
```

- To use the custom env later

```
source activate mynev   (# does not change your dot file
setup)
```
or
```
conda activate myenv    (# changes your dot file setup)
<...steps to use this conda env ... >
conda deactivate myenv
```

Parallel with Python

- Within a node
  - Use OpenMP-threaded math libs
  - Multiprocessing is OK too
- Multi-node parallelism
  - Best supported by mpi4py
  - Dask and PySpark frameworks also work
- Hybrid parallelism
  - Best route is mpi4py + threaded math libs
- Best to use shifter to scale up Python with mpi4py
  - https://docs.nersc.gov/programming/high-level-environments/python/scaling-up/#shifter-the-best-way-to-run-python-at-scale

What is Jupyter?

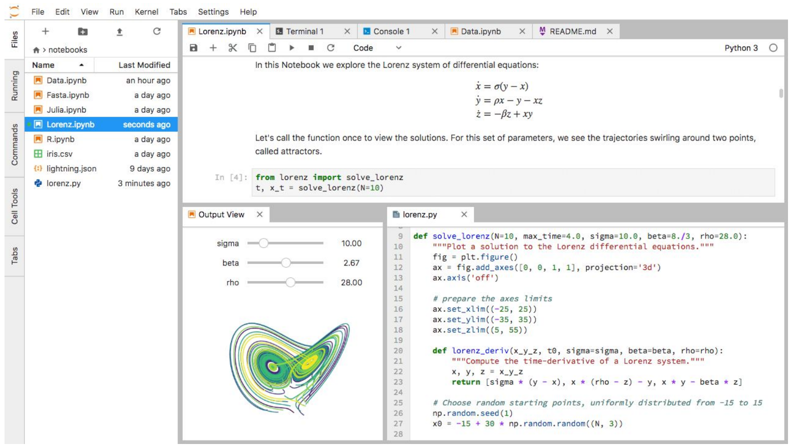
Interactive open-source web application

Allows you to <u>create</u> and <u>share</u> documents, “notebooks,” containing:

Live code
Equations
Visualizations
Narrative text
Interactive widgets

Things you can use Jupyter notebooks for:

Data cleaning and data transformation
Numerical simulation
Statistical modeling
Data visualization
Machine learning
Workflows and analytics frameworks



notebooks

Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

In this Notebook we explore the Lorenz system of differential equations:

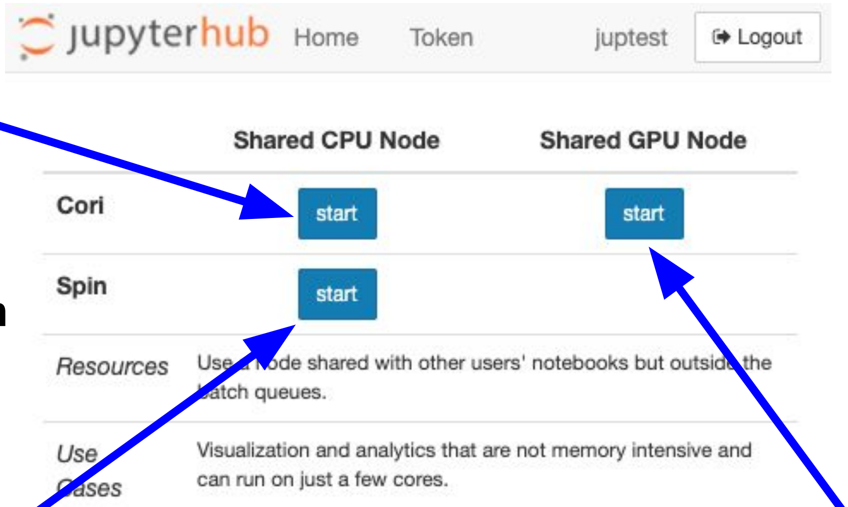
$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```
In [4]: from lorenz import solve_lorenz
        t, x_t = solve_lorenz(N=10)
```

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))

```

Which Notebook Server to Choose?



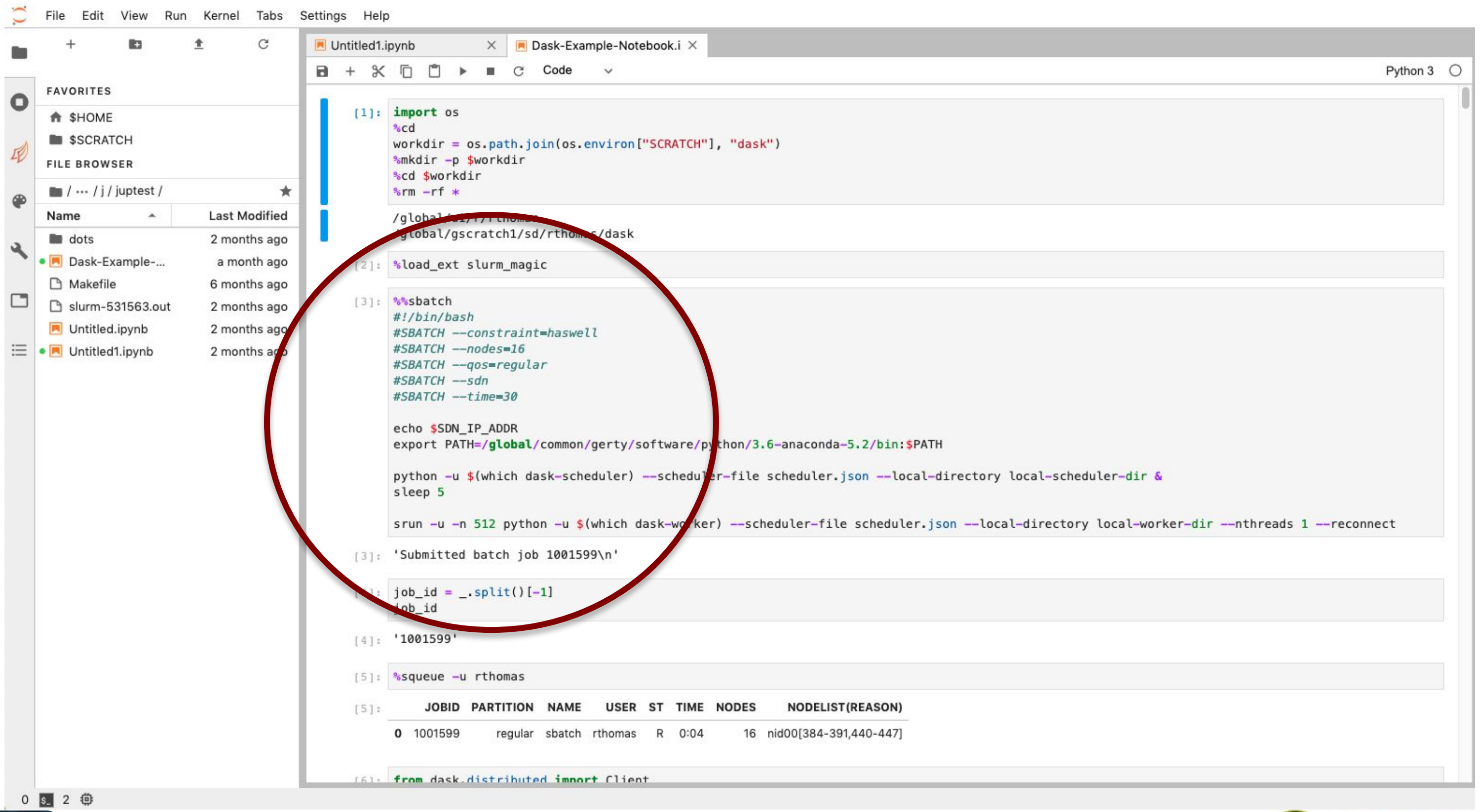
	Shared CPU Node	Shared GPU Node
Cori	start	start
Spin	start	
Resources	Use a node shared with other users' notebooks but outside the batch queues.	
Use Cases	Visualization and analytics that are not memory intensive and can run on just a few cores.	

Cori Shared CPU Node:
Notebook on cori{13,14,19}
Can see /cfs, $HOME, etc
Can see Cori $SCRATCH
Same Python env as ssh login
Can submit jobs via %sbatch

Spin Shared CPU Node:
External to Cori, in Spin
Can't see $SCRATCH
Can't run jobs
But *can* see /cfs, $HOME

Cori Shared GPU Node:
Notebook on cgpu{01-18}
Like Cori Shared CPU
Runs in a 4h job
Enabled if you have GPU QOS

JupyterLab Interface



Name	Last Modified
dots	2 months ago
Dask-Example-...	a month ago
Makefile	6 months ago
slurm-531563.out	2 months ago
Untitled.ipynb	2 months ago
Untitled1.ipynb	2 months ago

```
[1]: import os
     %cd
     workdir = os.path.join(os.environ["SCRATCH"], "dask")
     %mkdir -p $workdir
     %cd $workdir
     %rm -rf *

     /global/u1/r/rthomas
     /global/gscratch1/sd/rthomas/dask

[2]: %load_ext slurm_magic

[3]: %%sbatch
     #!/bin/bash
     #SBATCH --constraint=haswell
     #SBATCH --nodes=16
     #SBATCH --qos=regular
     #SBATCH --sdn
     #SBATCH --time=30

     echo $SDN_IP_ADDR
     export PATH=/global/common/gerty/software/python/3.6-anaconda-5.2/bin:$PATH

     python -u $(which dask-scheduler) --scheduler-file scheduler.json --local-directory local-scheduler-dir &
     sleep 5

     srun -u -n 512 python -u $(which dask-worker) --scheduler-file scheduler.json --local-directory local-worker-dir --nthreads 1 --reconnect

[3]: 'Submitted batch job 1001599\n'

[4]: job_id = _.split()[-1]
     job_id

[4]: '1001599'

[5]: %squeue -u rthomas

[5]:    JOBID  PARTITION   NAME     USER  ST  TIME  NODES  NODELIST(REASON)
     0  1001599    regular  sbatch  rthomas  R  0:04     16  nid00[384-391,440-447]

[6]: from dask.distributed import Client
```

BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY
Office of Science

Your Own Custom Jupyter Kernel

Most common Jupyter question:

"How do I take a conda environment and use it from Jupyter?"

Several ways to accomplish this, here's the easy one.

```
$ module load python
$ conda create -n myenv python=3.7
$ source activate myenv
(myenv) $ conda install ipykernel <other-packages>...
(myenv) $ python -m ipykernel install --user --name myenv-jupyter
```

Point your browser to jupyter.nersc.gov.
(You may need to restart your notebook server via control panel).
Kernel "myenv-jupyter" should be present in the kernel list.

NERSC Deep Learning Software Stack Overview

General strategy:

- Provide functional, performant installations of the most popular frameworks and libraries
- Enable flexibility for users to customize and deploy their own solutions

Frameworks:

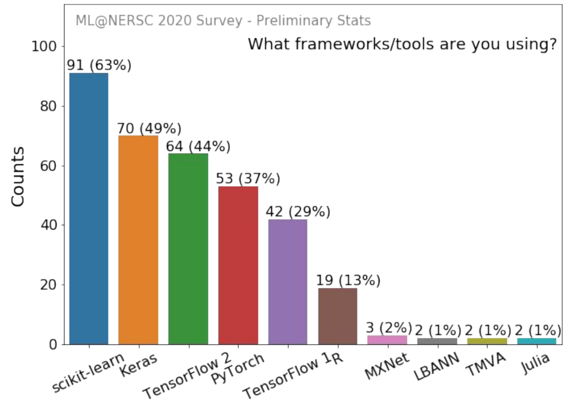
TensorFlow K Keras PyTorch

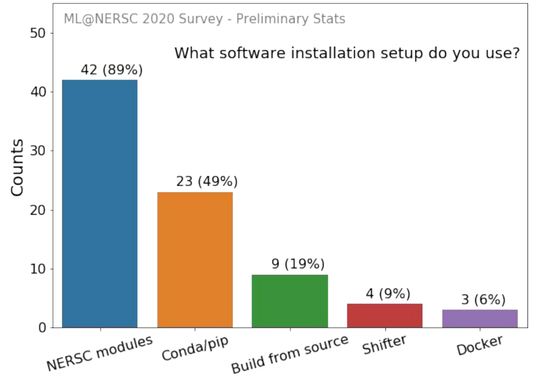
Distributed training libraries:

- Horovod
- PyTorch distributed
- Cray Plugin

Productive tools and services:

- Jupyter, Shifter





How to Use NERSC DL Software Stack

We have modules you can load which contain python and DL libraries:

```
module load tensorflow/intel-2.1.0-py37
module load pytorch/v1.5.0
```

Check which software versions are available with:

```
module avail tensorflow
```

You can install your own packages on top to customize:

```
pip install --user MY-PACKAGE
```

Or you can create your conda environments from scratch:

```
conda create -n my-env MY-PACKAGES
```

More on how to customize your setup can be found in the docs (TensorFlow, PyTorch).

We also have pre-installed Jupyter kernels.

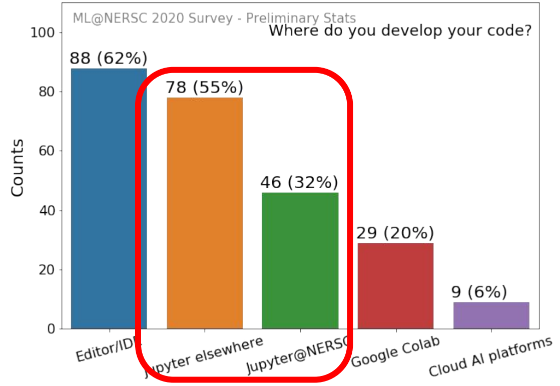
Jupyter for Deep Learning

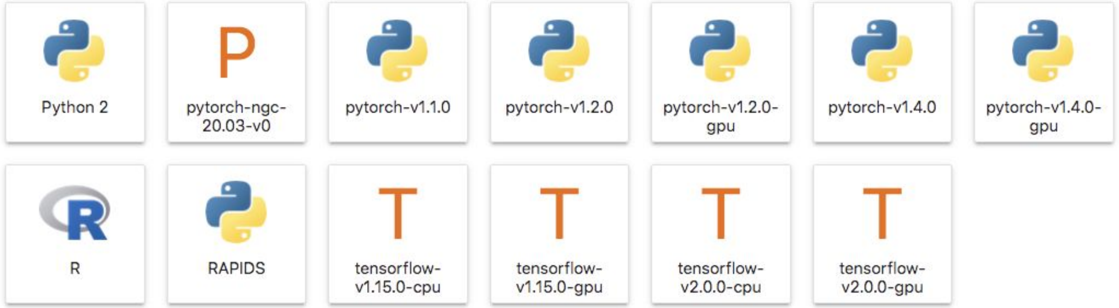
JupyterHub service provides a rich, interactive notebook ecosystem on Cori

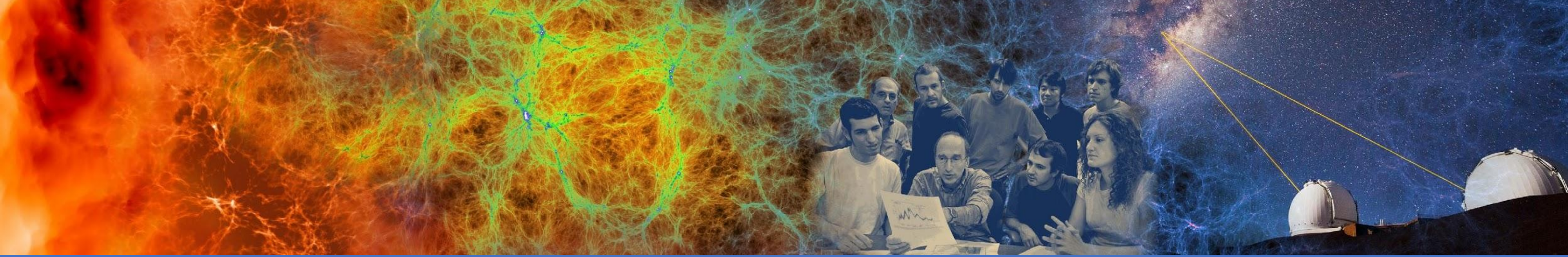
- Very popular service with hundreds of users
- A favorite way for users to develop ML code

Users can run their deep learning workloads

- on Cori CPU and Cori GPU
- using our pre-installed DL software kernels
- [using their own custom kernels]()



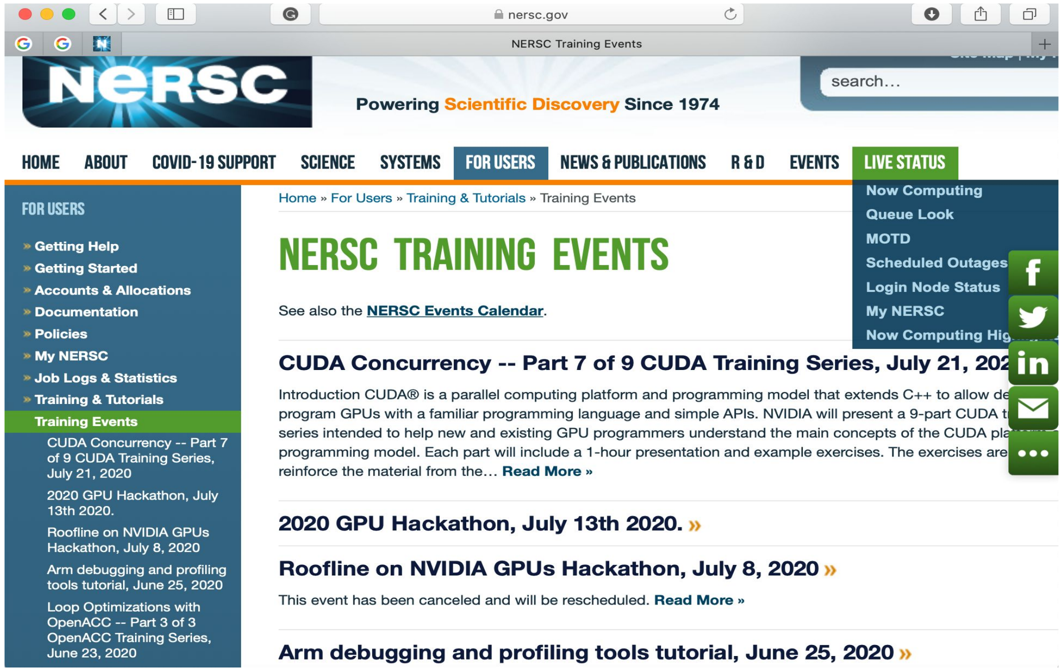




NERSC Online Resources

Online Resources: Classic NERSC Page

- https://www.nersc.gov
- Science, News, Publications
- Contact Us
- Live Status (MOTD): https://www.nersc.gov/live-status/motd/
- Training Events: https://www.nersc.gov/users/training/events/



Home » For Users » Training & Tutorials » Training Events

NERSC TRAINING EVENTS

See also the NERSC Events Calendar.

CUDA Concurrency -- Part 7 of 9 CUDA Training Series, July 21, 202

Introduction CUDA® is a parallel computing platform and programming model that extends C++ to allow de program GPUs with a familiar programming language and simple APIs. NVIDIA will present a 9-part CUDA t series intended to help new and existing GPU programmers understand the main concepts of the CUDA pla programming model. Each part will include a 1-hour presentation and example exercises. The exercises are reinforce the material from the... Read More »

2020 GPU Hackathon, July 13th 2020. »

Roofline on NVIDIA GPUs Hackathon, July 8, 2020 »

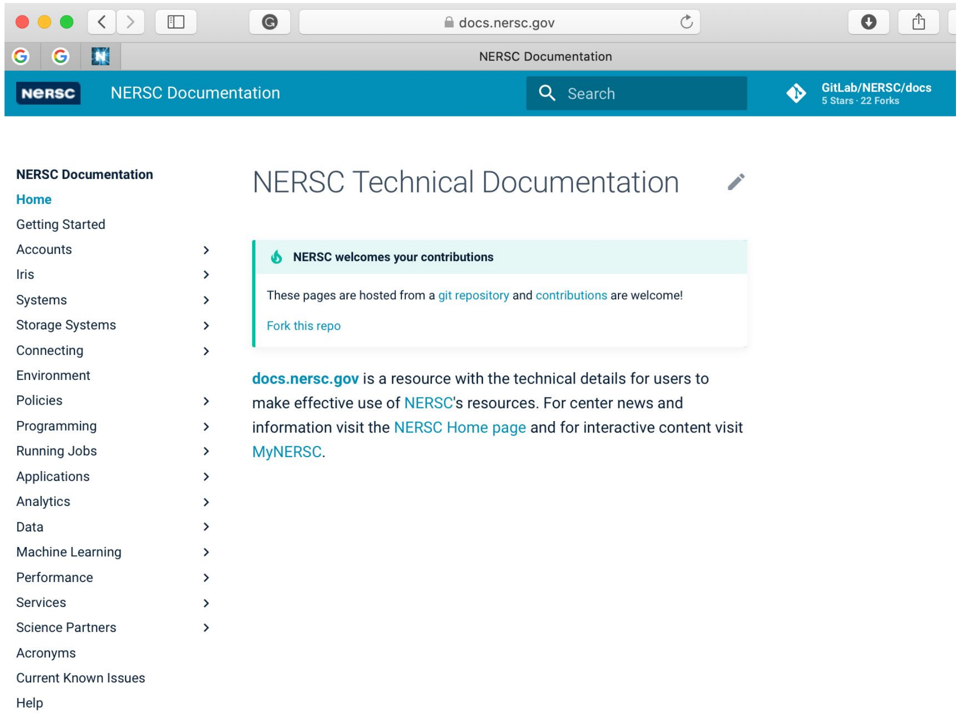
This event has been canceled and will be rescheduled. Read More »

Arm debugging and profiling tools tutorial, June 25, 2020 »

Online Resources: NERSC Docs

Technical Documentations
https://docs.nersc.gov

- Accounts
- IRIS
- Connecting
- Programming
- Running Jobs
- Applications
- Storage Systems
- Analytics
- Performance
- ...


docs.nersc.gov

NERSC Documentation

NERSC Documentation
Search
GitLab/NERSC/docs
5 Stars · 22 Forks

NERSC Documentation
Home
Getting Started
Accounts
Iris
Systems
Storage Systems
Connecting
Environment
Policies
Programming
Running Jobs
Applications
Analytics
Data
Machine Learning
Performance
Services
Science Partners
Acronyms
Current Known Issues
Help

NERSC Technical Documentation

NERSC welcomes your contributions

These pages are hosted from a git repository and contributions are welcome!

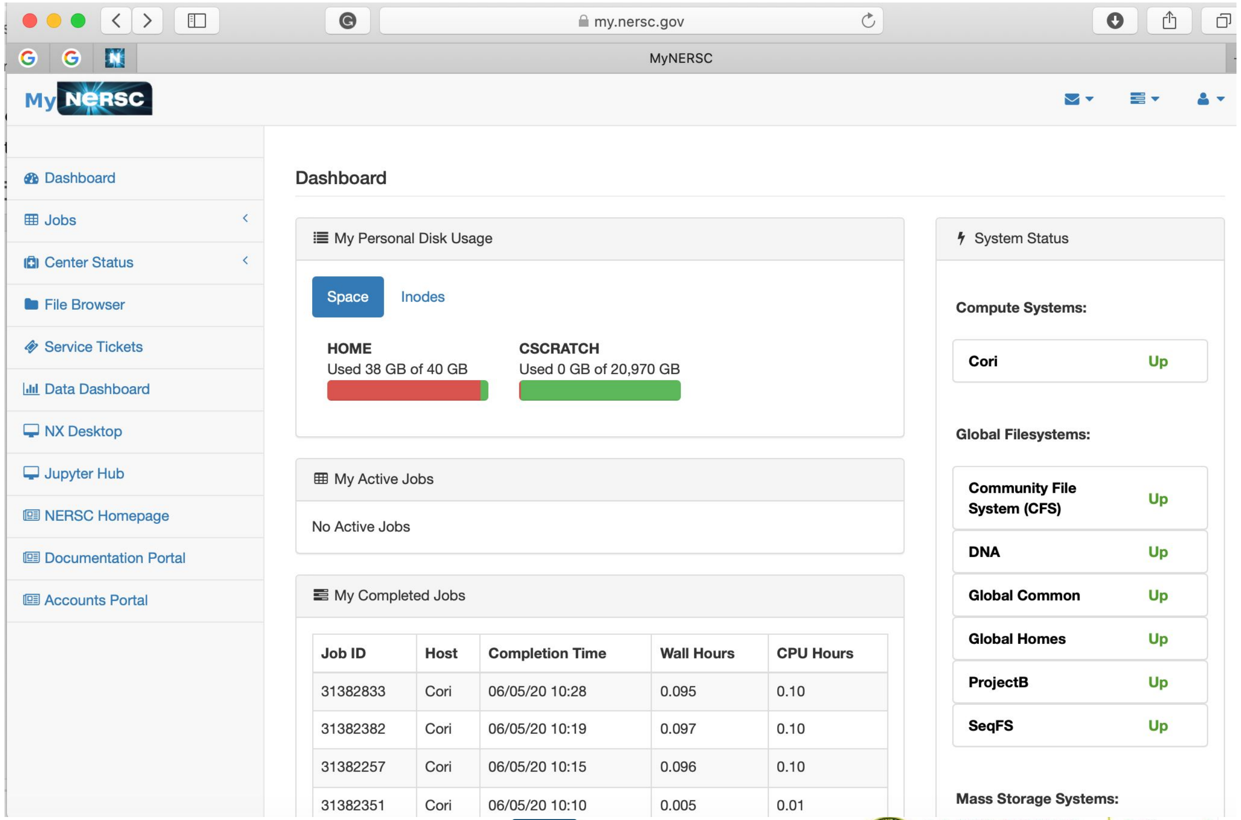
Fork this repo

docs.nersc.gov is a resource with the technical details for users to make effective use of NERSC's resources. For center news and information visit the NERSC Home page and for interactive content visit MyNERSC.

Online Resources: MyNERSC

https://my.nersc.gov

- Dashboard
- Jobs
- Center Status
- File Brwoser
- Service Tickets
- NX Desktop (disabled)
- Jupyter Hub
- Links to other useful pages



Dashboard

My Personal Disk Usage

HOME
Used 38 GB of 40 GB

CSCRATCH
Used 0 GB of 20,970 GB

My Active Jobs

No Active Jobs

My Completed Jobs

Job ID	Host	Completion Time	Wall Hours	CPU Hours
31382833	Cori	06/05/20 10:28	0.095	0.10
31382382	Cori	06/05/20 10:19	0.097	0.10
31382257	Cori	06/05/20 10:15	0.096	0.10
31382351	Cori	06/05/20 10:10	0.005	0.01

System Status

Compute Systems:

Cori	Up

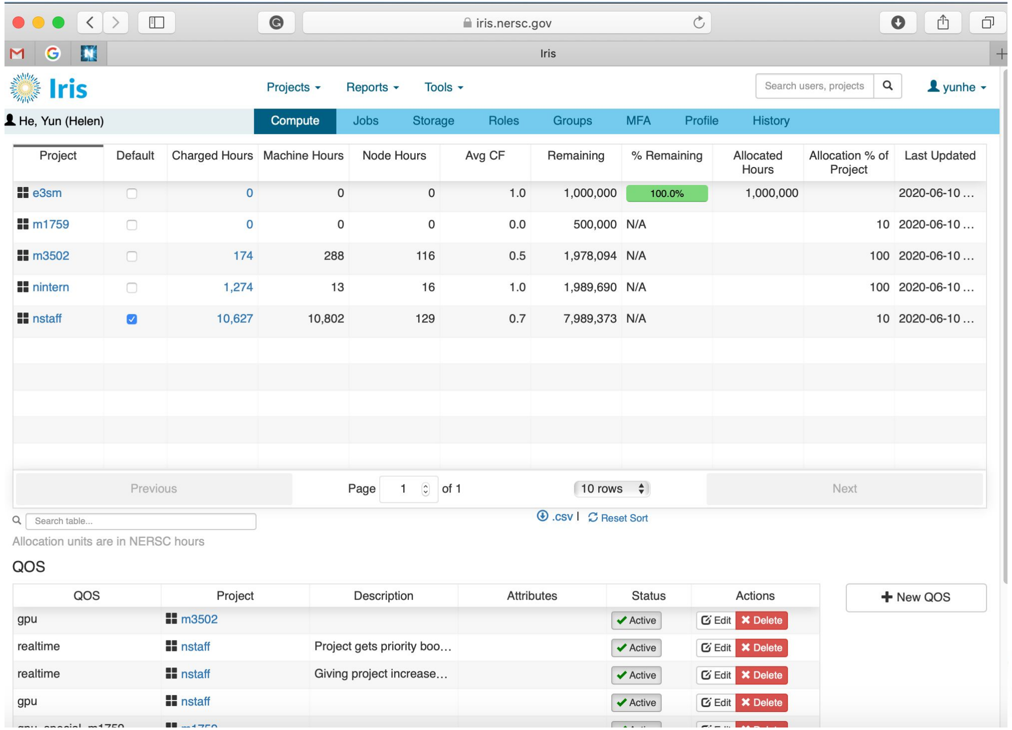
Global Filesystems:

Community File System (CFS)	Up
DNA	Up
Global Common	Up
Global Homes	Up
ProjectB	Up
SeqFS	Up

Mass Storage Systems:

Online Resources: IRIS

- IRIS: NERSC Account Management and Reporting: https://iris.nersc.gov
  - Change password
  - Change contact info
  - SSH Keys, MFA
  - Check usage info



iris.nersc.gov

Iris

Iris — Projects — Reports — Tools — Search users, projects — yunhe

He, Yun (Helen) — Compute — Jobs — Storage — Roles — Groups — MFA — Profile — History

Project	Default	Charged Hours	Machine Hours	Node Hours	Avg CF	Remaining	% Remaining	Allocated Hours	Allocation % of Project	Last Updated
e3sm	☐	0	0	0	1.0	1,000,000	100.0%	1,000,000		2020-06-10 ...
m1759	☐	0	0	0	0.0	500,000	N/A		10	2020-06-10 ...
m3502	☐	174	288	116	0.5	1,978,094	N/A		100	2020-06-10 ...
nintern	☐	1,274	13	16	1.0	1,989,690	N/A		100	2020-06-10 ...
nstaff	☑	10,627	10,802	129	0.7	7,989,373	N/A		10	2020-06-10 ...

Previous — Page 1 of 1 — 10 rows — Next

Search table... — .csv | Reset Sort

Allocation units are in NERSC hours

QOS

QOS	Project	Description	Attributes	Status	Actions
gpu	m3502			✓ Active	Edit Delete
realtime	nstaff	Project gets priority boo…		✓ Active	Edit Delete
realtime	nstaff	Giving project increase…		✓ Active	Edit Delete
gpu	nstaff			✓ Active	Edit Delete

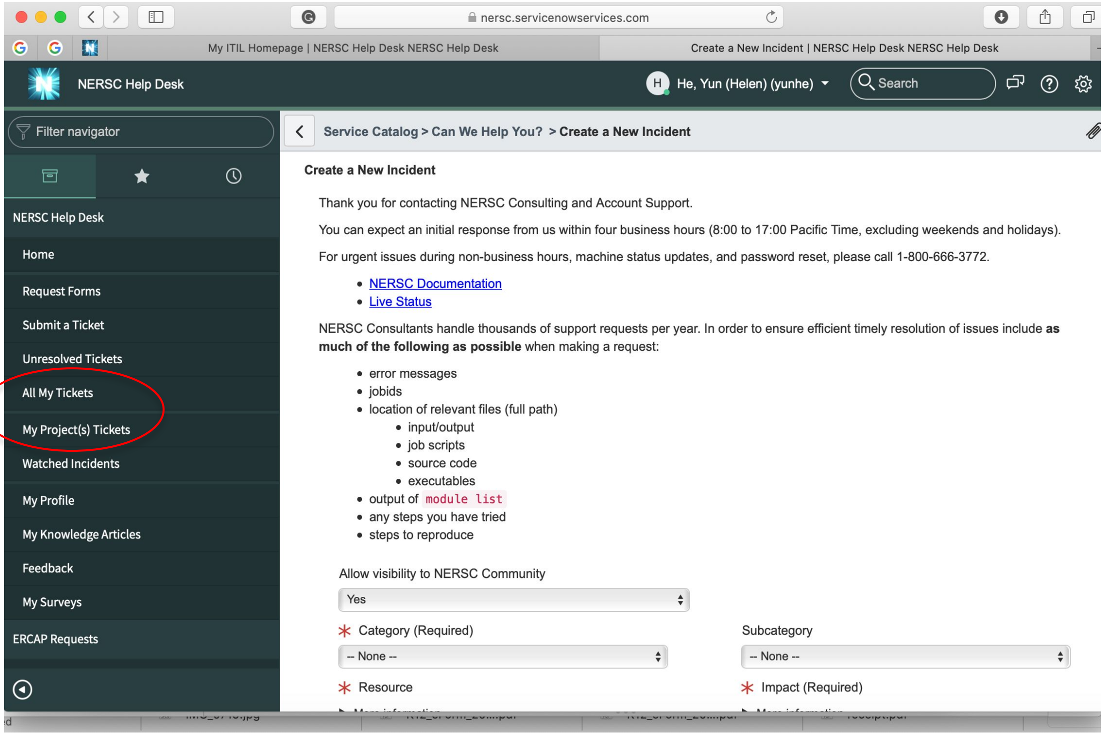
+ New QOS

NERSC

Online Resources: Help Portal

https://help.nersc.gov

- Submit tickets (ask questions)
- Request forms:
  - Quota Increase
  - Reservations
- Allocation (ERCAP) Requests



Create a New Incident

Thank you for contacting NERSC Consulting and Account Support.

You can expect an initial response from us within four business hours (8:00 to 17:00 Pacific Time, excluding weekends and holidays).

For urgent issues during non-business hours, machine status updates, and password reset, please call 1-800-666-3772.

- NERSC Documentation
- Live Status

NERSC Consultants handle thousands of support requests per year. In order to ensure efficient timely resolution of issues include **as much of the following as possible** when making a request:

- error messages
- jobids
- location of relevant files (full path)
  - input/output
  - job scripts
  - source code
  - executables
- output of `module list`
- any steps you have tried
- steps to reproduce

Allow visibility to NERSC Community: Yes

* Category (Required): -- None --

Subcategory: -- None --

* Resource

* Impact (Required)

https://my.nersc.gov Leads You to All Sites

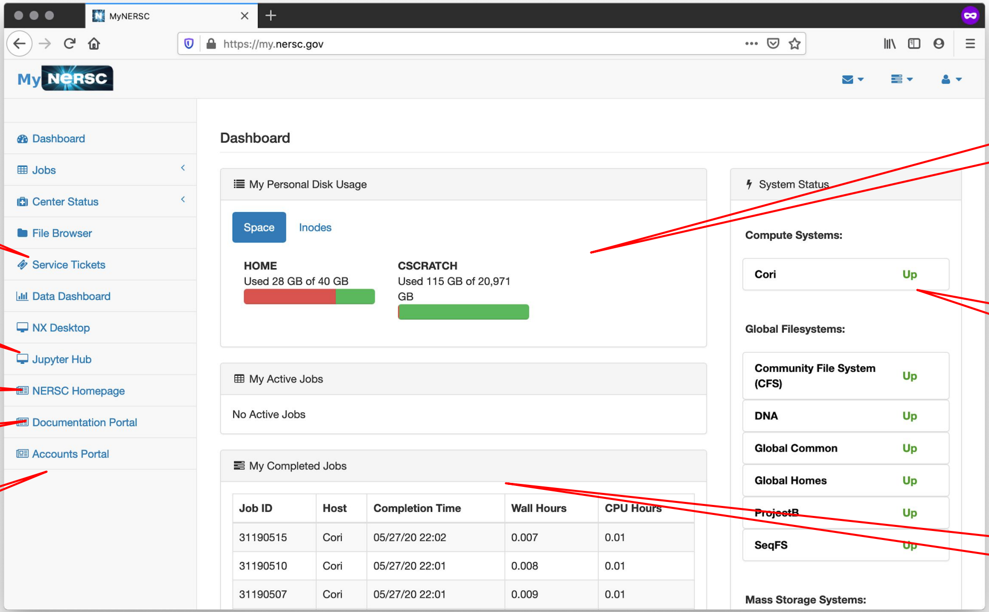
help.nersc.gov

jupyter.nersc.gov

www.nersc.gov

docs.nersc.gov

iris.nersc.gov



MyNERSC

https://my.nersc.gov

My NERSC

- Dashboard
- Jobs
- Center Status
- File Browser
- Service Tickets
- Data Dashboard
- NX Desktop
- Jupyter Hub
- NERSC Homepage
- Documentation Portal
- Accounts Portal

Dashboard

My Personal Disk Usage

Space Inodes

HOME
Used 28 GB of 40 GB

CSCRATCH
Used 115 GB of 20,971 GB

My Active Jobs

No Active Jobs

My Completed Jobs

Job ID	Host	Completion Time	Wall Hours	CPU Hours
31190515	Cori	05/27/20 22:02	0.007	0.01
31190510	Cori	05/27/20 22:01	0.008	0.01
31190507	Cori	05/27/20 22:01	0.009	0.01

System Status

Compute Systems:

Cori	Up

Global Filesystems:

Community File System (CFS)	Up
DNA	Up
Global Common	Up
Global Homes	Up
ProjectB	Up
SeqFS	Up

Mass Storage Systems:

my disk quota

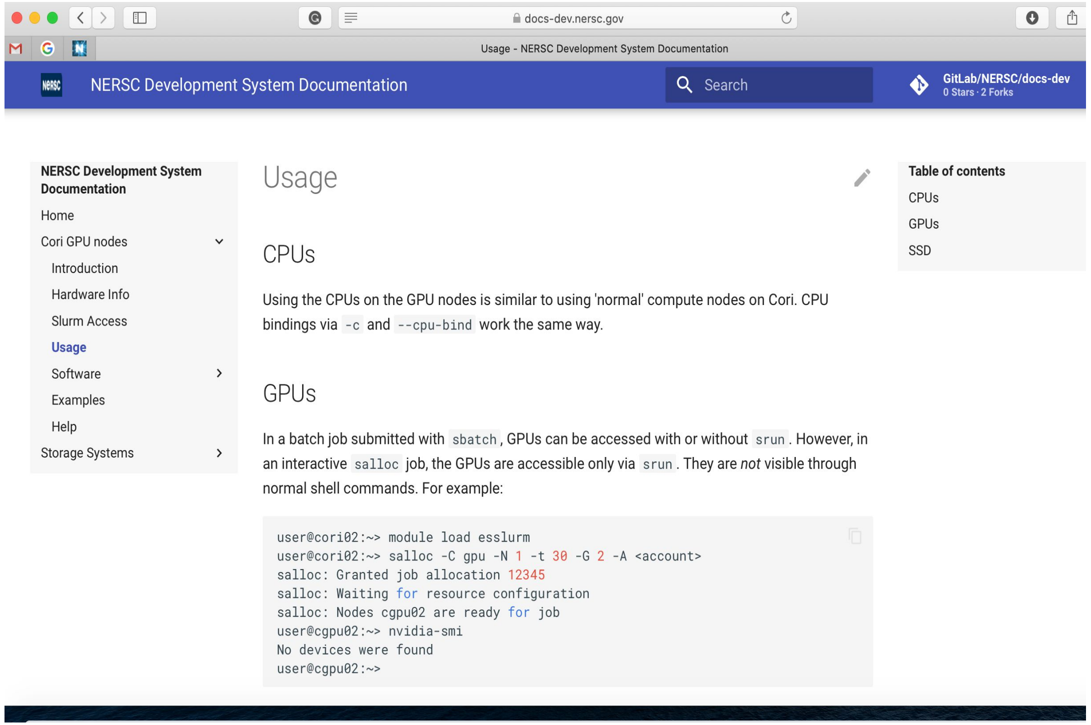
is cori up?

my jobs

Online Resources: Cori GPU Documentation

https://docs-dev.nersc.gov

- GPU nodes
  - Hardware info
  - Slurm access
  - Usage
  - Software
    - Compilers
    - Math libraries
    - Python
    - Shifter
    - Profiling
  - Examples



Usage

CPUs

Using the CPUs on the GPU nodes is similar to using 'normal' compute nodes on Cori. CPU bindings via `-c` and `--cpu-bind` work the same way.

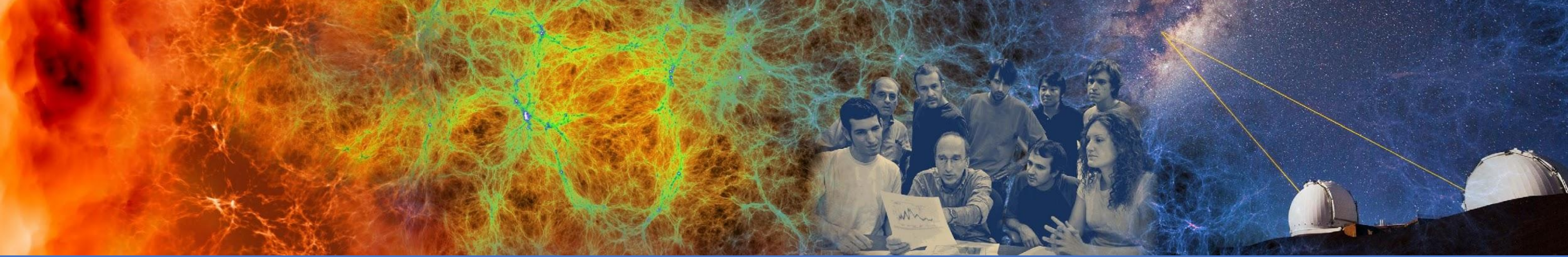
GPUs

In a batch job submitted with `sbatch`, GPUs can be accessed with or without `srun`. However, in an interactive `salloc` job, the GPUs are accessible only via `srun`. They are *not* visible through normal shell commands. For example:

```
user@cori02:~> module load esslurm
user@cori02:~> salloc -C gpu -N 1 -t 30 -G 2 -A <account>
salloc: Granted job allocation 12345
salloc: Waiting for resource configuration
salloc: Nodes cgpu02 are ready for job
user@cgpu02:~> nvidia-smi
No devices were found
user@cgpu02:~>
```

Acknowledgement

- Used / Adapted some slides and materials from the upcoming NERSC New user training (June 16, 2020)
    - Thanks Rebecca Hartman-Baker, Clayton Bagwell, Steve Leak, Zhengji Zhao, Woo-Sun Yang, Bill Arndt, Wahid Bhimji, Lisa Gerhardt, Quincy Koziol, Laurie Stephey, Rollin Thomas, Shane Canon, Mustafa Mustafa
- https://www.nersc.gov/users/training/events/new-user-training-june-16-2020/
    - You are encouraged to attend the all day training next Tuesday, or join the particular sessions of interest for in-depth understanding.



Hands-on Exercises

Hands-on Exercises

- % cd $SCRATCH
- % cp -r /global/cfs/cdirs/training/2020/CSSS .
- % cd CSSS
- Beginner users follow: run-hello.README
- Advanced users follow: run-xthi.README
- References
  - Running Jobs: https://docs.nersc.gov/jobs/
  - Interactive Jobs: https://docs.nersc.gov/jobs/examples/#interactive

Thank You